

MSIM4301
Edisi 1

MODUL 01

Konsep Pemrograman Berorientasi Objek

Dr. Heni Jusuf, S.Kom., M.Kom.

Daftar Isi

| | |
|--|-------------|
| Modul 01 | 1.1 |
| Konsep Pemrograman Berorientasi Objek | |
| Kegiatan Belajar 1 | 1.5 |
| Pengantar Pemrograman Berorientasi Objek | |
| Latihan | 1.15 |
| Rangkuman | 1.15 |
| Tes Formatif 1 | 1.16 |
| Kegiatan Belajar 2 | 1.19 |
| Instalasi Program Java | |
| Latihan | 1.37 |
| Rangkuman | 1.37 |
| Tes Formatif 2 | 1.38 |
| Kunci Jawaban Tes Formatif | 1.41 |
| Glosarium | 1.42 |
| Daftar Pustaka | 1.43 |



Pendahuluan

Pemrograman Berorientasi Objek (PBO) atau *Object Oriented Programming* (OOP) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari PBO/OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, sedangkan objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Misalkan diambil contoh Pesawat, sebagai sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dan lain-lain. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, dimana objek-objek tersebut saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

Pada Modul 1 ini, Anda akan dibekali dengan penjelasan tentang pentingnya pemrograman berorientasi objek, perbedaan pemrograman prosedural dengan pemrograman berorientasi objek, dan melakukan *install* program java.

Setelah mempelajari modul 1 ini, diharapkan mahasiswa dapat

1. membedakan pemrograman berorientasi objek dengan pemrograman prosedural,
2. mengidentifikasi *compiler*, *interpreter* dan *editor*,
3. melakukan *install* program java dengan benar.

Pembahasan dalam Modul 1 ini terdiri atas 2 kegiatan belajar, yaitu tentang:

1. Konsep Pemrograman Berorientasi Objek.
2. *Instalasi* Program Java.

Untuk itu Anda diharapkan mengikuti petunjuk belajar sebagai berikut.

1. **Bacalah** bagian **uraian dan contoh** dari setiap Kegiatan Belajar dengan baik sampai dimengerti, dipahami, dan diterapkan.
2. **Kerjakan Latihan** dengan baik dan jujur, penuh kesungguhan dan tanggung jawab.
3. **Bacalah Rangkuman** yang disediakan untuk memberikan ringkasan pemahaman tentang Konsep Pemrograman Berorientasi Objek.
4. **Kerjakan Tes Formatif** yang disediakan untuk mengecek seberapa jauh Anda mencapai tujuan pembelajaran setiap kegiatan belajar tanpa melihat rambu-rambu jawaban yang tersedia.
5. **Bacalah Glosarium** yang disediakan untuk menyamakan persepsi tentang istilah yang dipakai dalam Konsep Pemrograman Berorientasi Objek.

6. **Jika** Anda merasa telah menjawab Tes Formatif dengan baik dan benar, bandingkanlah jawaban tersebut dengan kriteria yang tersedia. Apabila setelah dihitung ternyata Anda telah mencapai tingkat penguasaan minimal 80%, Anda dapat meneruskan ke Kegiatan Belajar Berikutnya.

Kegiatan
Belajar

1

Pengantar Pemrograman Berorientasi Objek

Dasar untuk Pemrograman Berorientasi Objek (PBO) atau *Object Oriented Programming* (OOP) dimulai pada awal tahun 1960-an. Sebuah terobosan yang melibatkan contoh dan objek telah ditemukan di MIT dengan PDP-1, dan untuk pertama kalinya digunakan bahasa pemrograman objek yaitu Simula 67. Bahasa Simula 67 dikembangkan oleh Kristen Nygaard dan Ole-Johan Dahl dari Norwegia dan dirancang untuk tujuan membuat simulasi..

Bahasa Simula 67 digunakan untuk simulasi *exploding* kapal, karena dapat membuat sebuah grup yang berbeda pada bagian kapal ke sebuah kategori. Setiap jenis bagian kapal akan memiliki *class*, dan *class* akan menghasilkan perilaku yang unik dan data. Simula 67 tidak hanya bertanggung jawab untuk memperkenalkan konsep *class*, tetapi juga memperkenalkan *instance* dari *class*.

Istilah *Object Oriented Programming* (OOP) pertama kali digunakan oleh Xerox PARC di luar bahasa pemrograman lainnya. Istilah ini digunakan untuk merujuk kepada proses yang menggunakan *object* sebagai dasar untuk penghitungan. Tim *Smalltalk* yang telah terinspirasi oleh proyek Simula 67, mereka telah merancang *Smalltalk* sehingga menjadi dinamis. *Object* dapat diubah, diciptakan, atau dihapus, dan hal ini berbeda dengan sistem statis yang digunakan secara umum. *Smalltalk* juga merupakan bahasa pemrograman yang pertama kali memperkenalkan konsep *inheritance*. *Inheritance* adalah fitur dimana sebuah *object* dapat mempunyai *object* turunan. Konsep *inheritance* menjadikan *Smalltalk* melebihi Simula 67 dan pemrograman sistem analog, meskipun sistem yang canggih pada saat itu belum bisa menggunakan konsep *inheritance* tersebut .

Simula 67 merupakan sistem yang memberikan inspirasi banyak bahasa pemrograman lain, termasuk Pascal dan Lips. Pada tahun 1980-an, pemrograman berorientasi objek telah menjadi dominan, dan faktor utama dalam hal ini yaitu C++. PBO juga penting untuk pengembangan *grafik user interface*. *Cocoa* dengan struktur yang ada dalam *Mac OS X* adalah salah satu contoh dari GUI yang dinamis yang bekerja dengan bahasa pemrograman berorientasi *object*. Paradigma dari pemrograman ini juga telah memainkan peranan penting dalam pengembangan *event-driven programming*.

Niklaus Wirth dan asosiasinya membuat konsep seperti *modular programming* dan *abstraction* dan mengembangkan sistem yang menghimpun dua unsur ini. Kedua sistem ini adalah *Oberon* dan *Modula-2*. *Oberon* menggunakan pendekatan yang unik

untuk *class* dan *object oriented* yang jauh berbeda dibandingkan C++ atau *Smalltalk*. Sejak diperkenalkannya PBO, sejumlah besar bahasa pemrograman modern sekarang menggunakan konsep ini. Beberapa di antaranya adalah Visual basic, Java, C, dan C++. Ada beberapa masalah kompatibilitas, karena banyak bahasa pemrograman yang tidak dirancang dengan pendekatan PBO. Pemrograman bahasa berorientasi objek yang murni tidak memiliki banyak *method* yang diperlukan oleh *programmer*.

Untuk memecahkan masalah tersebut, sejumlah peneliti telah mencoba membuat desain baru untuk bahasa pemrograman yang menggunakan konsep *object oriented* yang masih tetap menggunakan *method* yang diperlukan oleh *programmer*. Satu contoh dari sebuah bahasa pemrograman yang telah dicapai adalah *Eiffel*. Bahasa pemrograman lain yang telah berusaha untuk memecahkan masalah ini adalah Java. Java menjadi populer karena menggunakan *virtual Machine*, dan sangat mirip dengan C++ dan C. Mesin *virtual* penting karena memungkinkan kode untuk dijalankan pada berbagai *platform* tanpa perlu diubah. Sistem lain yang serupa adalah *Microsoft NET*. Banyak pengembang sekarang memahami pentingnya PBO, dan menggunakannya dalam pembuatan berbagai aplikasinya masing-masing. Banyak peneliti terus membuat pengembangan dengan menggunakan pendekatan *object oriented*.

A. PENTINGNYA PEMROGRAMAN BERORIENTASI OBJEK

Metode pemrograman dalam dunia *programming* terbagi atas 5 jenis, yaitu: (1) Imperatif, (2) Terstruktur, (3) Prosedural, (4) Objek Oriented, dan (5) Fungsional. Sebelum membahas tentang Pemrograman Berorientasi Objek akan dibahas terlebih dahulu tentang Pemrograman Terstruktur dan Pemrograman Prosedural.

Pemrograman Terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program. Selain pengertian diatas, Pemrograman Terstruktur juga dapat diartikan sebagai suatu aktivitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis, dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami. Prinsip dari pemrograman terstruktur adalah “Jika suatu proses telah sampai pada suatu titik/langkah tertentu, maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya/kembali lagi ke baris sebelumnya, kecuali pada langkah – langkah untuk proses berulang (*Loop*)”.

Pemrograman Terstruktur memiliki sifat-sifat sebagai berikut.

1. Memuat teknik pemecahan masalah yang logis dan sistematis.
2. Memuat algoritma yang efisien, efektif, dan sederhana.
3. Program disusun dengan logika yang mudah dipahami.
4. Tidak menggunakan perintah *GO-TO*.
5. Biaya pengujian program relatif rendah.
6. Memiliki dokumentasi yang baik.
7. Biaya perawatan dan dokumentasi yang dibutuhkan relatif rendah.

Pemrograman Prosedural pada dasarnya adalah metode pemrograman yang berupa baris-baris perintah yang dieksekusi secara urut mulai dari baris atas hingga bawah. Pada model Pemrograman tradisional atau prosedural (disebut *process-oriented model*) ini, semua data dan kode digabung menjadi satu bagian dalam satu program. Untuk program – program sederhana yang hanya membutuhkan beberapa buah baris kode, penggunaan model ini tentu tidak begitu menjadi masalah. Contoh beberapa Bahasa pemrograman Prosedural adalah: Cobol Turbo Prolog, C, Pascal, Delphi, Borland Delphi.

Pemrograman Prosedural memiliki kelebihan-kelebihan, yaitu:

1. memiliki algoritma pemecahan masalah yang sederhana, standar, dan efektif,
2. penulisan program memiliki struktur logika yang benar dan mudah dipahami,
3. program hanya terdiri dari 3 (tiga) struktur dasar, yaitu struktur berurutan struktur seleksi, dan struktur perulangan,
4. memiliki dokumentasi yang baik,
5. menghindari penggunaan pernyataan *GO TO*, yang akan menjadikan program tidak terstruktur dengan baik.

Sedangkan kekurangan dari Pemrograman Prosedural adalah:

1. program cukup sulit untuk proses perawatan,
2. *method* yang tersedia sulit untuk diubah tanpa harus mempengaruhi *method* sistem secara keseluruhan,
3. butuh usaha yang keras untuk menterjemahkan *Business Models* dalam *programming models*,
4. mungkin dapat bekerja dengan baik pada saat terisolasi tetapi tidak pada saat terintegrasi dengan sistem lain.

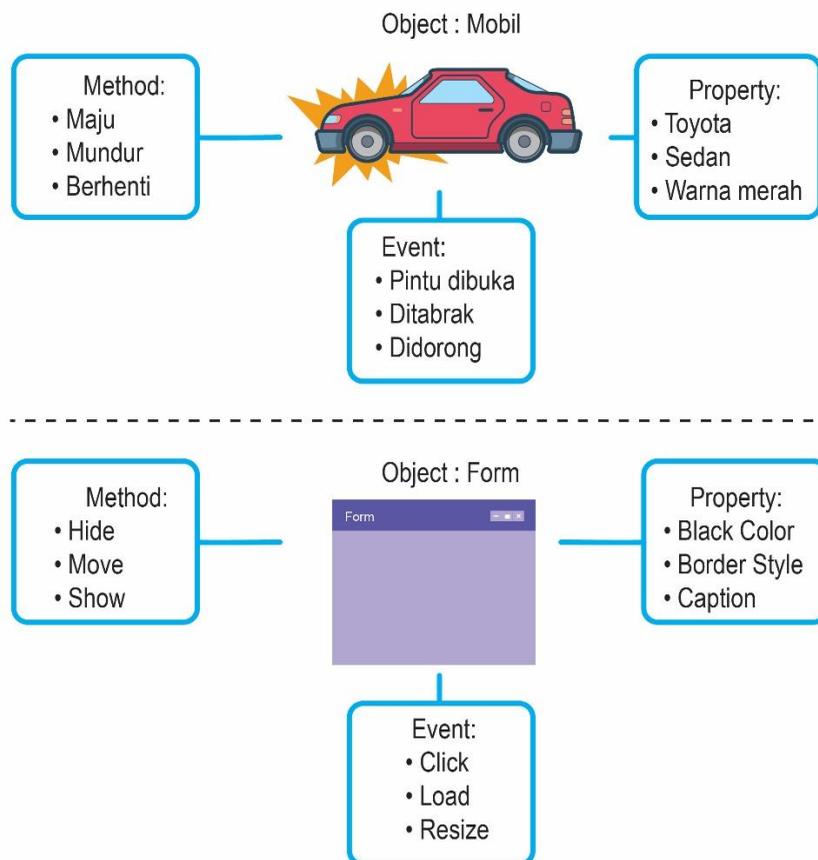
OOP (*Object Oriented Programming*) atau dalam bahasa Indonesia dikenal dengan Pemrograman Berorientasikan Objek (PBO) merupakan sebuah paradigma atau metode pemrograman yang berorientasikan objek. PBO mendefinisikan semua data dan *method* ke dalam beberapa *class* atau objek-objek agar bisa saling bekerja sama dalam memecahkan masalah. Model data berorientasi objek memberikan lebih banyak fleksibilitas, seperti kemudahan dalam mengubah program, dan secara luas bisa digunakan dalam pemrograman skala besar.

Kelebihan dari metode PBO sendiri adalah sebagai berikut.

1. *Maintenance* program lebih mudah dan program yang dibuat dapat lebih mudah dibaca dan dipahami. Selain itu, PBO dapat mengontrol kerumitan program hanya dengan cara mengizinkan perincian-perincian yang dibutuhkan oleh programmer.

2. Mudah dalam perubahan program, bisa berupa penambahan atau penghapusan fitur atau objek tertentu. Contoh perubahan yang bisa dilakukan antara lain penambahan dan penghapusan data di dalam suatu *database*.
3. Objek-objek di dalam program dapat digunakan sesering mungkin oleh *programmer* untuk menyimpan objek-objek yang dirancang ke dalam sebuah *module*, yang dapat disisipkan ke dalam baris kode baru. Penambahan bisa dilakukan dengan sedikit perubahan atau tanpa perubahan pada kode program utama.

Pada PBO, *method* dan variabel dibungkus dalam sebuah objek atau *class* yang dapat saling berinteraksi, sehingga membentuk sebuah program. Variabel dalam objek akan menyimpan data dari objek, sedangkan *method* akan menentukan operasinya. Untuk lebih jelasnya pada Gambar 1.1 berikut ini.



Gambar 1.1
Konsep PBO

Contoh objek dalam dunia nyata: mobil, burung, *drone*, meja, pohon, dan lainnya.

```
-----OBJEK
Drone
-----Variabel/Atribut
energi = 100;
ketinggian = 200;
kecepatan = 29;
-----Fungsi
terbang();
matikanMesin();
turun();
maju();
mundur();
belok();
-----
```

Semua objek di dunia nyata yang memiliki sifat dan tingkah laku, bisa kita representasikan dalam kode.

Kata kunci yang perlu diingat:

“Objek isinya data dan *method*”

Sebelum mempelajari lebih jauh tentang PBO, sebaiknya kita mengetahui terlebih dahulu istilah – istilah dalam PBO berikut ini.

- a. *Class*, adalah kerangka dasar dari objek yang akan diciptakan, bisa berupa struktur yang mendefinisikan data atau *method* dari objek. Contoh penamaan *class* adalah: motor, laptop, anggota, dan lainnya.
- b. *Property*, adalah data yang dimiliki oleh *class*. Contohnya pada *class* Motor, memiliki *property* sebagai berikut.
 - 1) Tipe.
 - 2) Warna.
 - 3) Produsen.
- c. *Method*, adalah perilaku dari sebuah *class*. Bisa juga disebut sebagai tindakan yang bisa dilakukan oleh suatu *class*.
- d. Contoh *method-method* pada *class* motor, antara lain adalah sebagai berikut.
 - 1) *Start, method* untuk menjalankan motor,
 - 2) *Stop, method* untuk menghentikan laju motor,
 - 3) *Ganti Gigi, method* untuk ganti gigi, dan
 - 4) *Turn, method* untuk belok kiri atau kanan.
- e. Variabel *class*, merupakan variabel yang dibagikan oleh semua turunan dari *class*. Variabel *class* didefinisikan di dalam *class*, tapi di luar *method* yang ada di dalam *class* tersebut.
- f. Data *member*, merupakan variabel penyimpan data yang berhubungan dengan *class* dan objek.

- g. *Overloading Method*, merupakan *method* yang namanya sama di dalam *class*, tetapi jumlah dan tipe argumennya berbeda, sehingga bisa dilakukan beberapa hal yang berbeda.
- h. *Overloading Operator*, merupakan pembuatan beberapa *method* untuk satu operator.
- i. Variabel Instansiasi, merupakan variabel yang didefinisikan di dalam suatu *method* dan hanya menjadi milik dari satu *class*.
- j. Pewarisan, merupakan pewarisan karakteristik dari sebuah *class* kepada *class* lain yang menjadi turunannya.
- k. Instansiasi, merupakan pembuatan objek dari suatu *class*.

PBO dikenal juga sebagai metode pemrograman modern yang lebih efisien dan banyak digunakan pada *Framework*.

B. PERBEDAAN PEMROGRAMAN PROSEDURAL DENGAN PEMROGRAMAN BERORIENTASI OBJEK

Untuk membedakan antara pemrograman prosedural dan pemrograman berorientasi objek bisa menggunakan kriteria-kriteria yang ada pada Tabel 1.1.

Tabel 1.1
Perbedaan Antara Pemrograman Prosedural dengan PBO

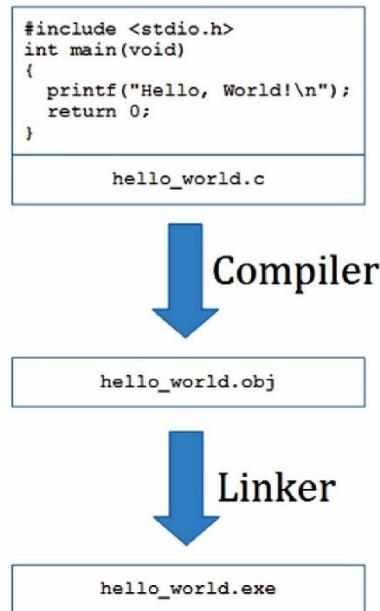
| Prosedural | PBO |
|---|---|
| Diselesaikan dalam bentuk prosedur atau <i>method</i> Program merupakan urutan-instruksi | <i>Method</i> dan data menjadi satu kesatuan yang disebut objek |
| Program dipecah-pecah ke dalam subprogram yang lebih sederhana | Objek-objek dalam PBO bersifat aktif |
| Fokus utama pada prosedur dan <i>method</i> <i>Method</i> dan prosedur digunakan untuk memanipulasi data | Cara pandang: program bukan merupakan urutan-instruksi saja, tapi diselesaikan oleh objek-objek yang bekerja sama untuk menyelesaikan masalah |
| Data bersifat pasif | |

1. Mengidentifikasi *Compiler*, *Interpreter* dan *Editor*

Compiler adalah sebuah perangkat lunak (*software*) yang menerjemahkan kode sumber (*source code*) sebuah program komputer ke dalam kode objek (*object code*) atau file biner. Jadi *compiler* menghasilkan file baru, yaitu file objek atau file biner dari file kode sumber. Program yang dibangun oleh *compiler* berjalan lebih cepat dibanding program yang di produksi oleh *interpreter*, di samping itu juga bersifat *independent*. Contoh bahasa pemrograman yang menggunakan *compiler* adalah *Visual Basic*, *Visual Delfi*, dan *Pascal*.

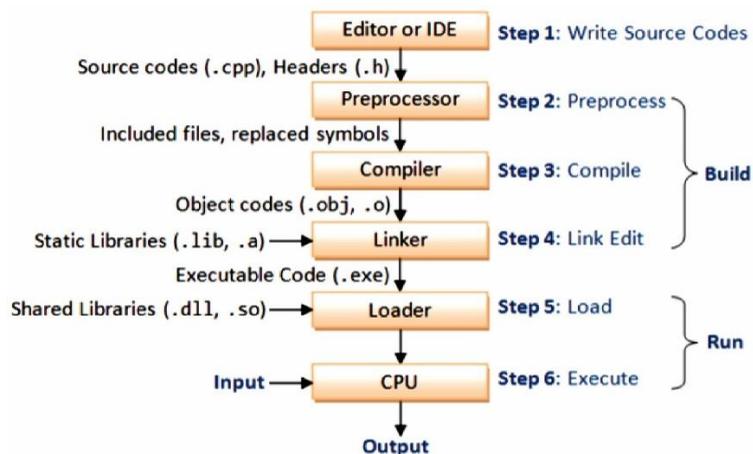
Secara umum tahap-tahap kompilasi dapat dilihat pada Gambar 1.2, dengan penjelasan berikut :

- membaca *source code* program yang ditulis dari memori komputer,
- mengubah *source code* tersebut menjadi *object code* (bahasa *assembly*).
- menghubungkan (*linked*) *object code* dengan *library* yang dibutuhkan untuk membentuk *file* yang bisa dieksekusi.



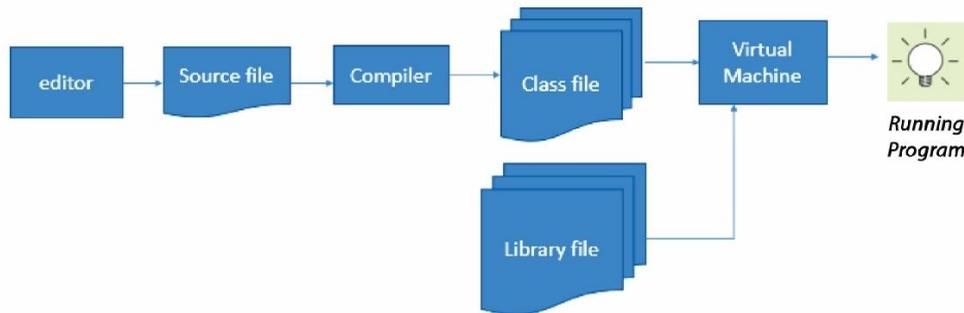
Gambar 1.2
Tahap Kompilasi

Adapun tahapan proses kompilasi yang dilakukan oleh *compiler* dapat dilihat pada Gambar 1.3 berikut.



Gambar 1.3
Urutan Proses *Compiler*

Untuk dapat menampilkan hasil, baik itu hasil yang sesuai maupun hasil yang salah diperlukan urutan kode program yang akan dieksekusi, seperti terlihat pada Gambar 1.4.



Gambar 1.4
Urutan *Source Code* ke *Running Program*

Interpreter adalah perangkat lunak yang mampu mengeksekusi *code* program (yang ditulis oleh *programmer*) lalu menterjemahkannya ke dalam bahasa mesin, sehingga mesin melakukan instruksi yang diminta oleh programmer. Perintah-perintah yang dibuat oleh *programmer* dieksekusi baris demi baris, sambil mengikuti logika yang terdapat di dalam kode tersebut. *Interpreter* akan menterjemahkan kode sumber ke dalam kode sasaran secara langsung atau *orally*, kemudian *translator* akan menerjemahkan kode sumber ke kode sasaran secara tertulis. Java dijalankan menggunakan *interpreter* yaitu *Java Virtual Machine (JVM)*. Hal ini menyebabkan *source code Java* yang telah dikompilasi menjadi *Java bytecodes* dapat dijalankan pada *platform* yang berbeda-beda.

Proses pada *interpreter* sangat berbeda dengan *compiler*, dimana pada *compiler*, hasilnya sudah langsung berupa satu kesatuan perintah dalam bentuk bahasa mesin, yang proses penterjemahan dilaksanakan sebelum program tersebut dieksekusi. Program dalam bahasa mesin adalah kumpulan kode biner yang merupakan instruksi yang bisa dijalankan oleh komputer atau sering disebut sebagai kode objek.

Bahasa *assembly* adalah sebuah *software* yang terdiri dari instruksi-instruksi yang menggantikan kode-kode *biner* dari bahasa mesin dengan “*mnemonik*” yang mudah diingat. Misalkan sebuah instruksi penambahan dalam bahasa mesin dengan kode “10110011” yang dalam bahasa *assembly* dapat dibuat dalam instruksi *mnemonik ADD*, sehingga mudah diingat dibandingkan dengan angka 0 dan 1. Dalam setiap instruksi bahasa *assembly* membutuhkan suatu *operand*, baik berupa data langsung maupun suatu lokasi memori yang menyimpan data yang bersangkutan. Bahasa *assembly* sering juga disebut kode sumber atau kode simbolik yang tidak dapat dijalankan oleh prosesor, sedangkan *assembler* adalah *software* yang dapat menterjemahkan program bahasa *assembly* ke program bahasa mesin.

Dalam tahapan *compile*, terdapat *Linker* yaitu *software* yang menerjemahkan program objek (berekstensi *.obj*) ke bentuk program tereksekusi (berekstensi *.exe* atau *.com*), sedangkan untuk membuat *file object* ke bentuk file yang dapat dieksekusi (berekstensi *.com* atau *.exe*) bisa digunakan file *TLINK.exe*.

Dalam Java terdapat editor yang digunakan untuk menuliskan *source code* bahasa pemrograman Java. Editor tersebut harus mampu menyimpan file dengan ekstensi *.java* yang merupakan ekstensi dari file untuk menyimpan *source code* bahasa pemrograman Java. Terdapat dua kelompok editor dalam Java, yaitu *Text editor* dan *Integrated Development Environment (IDE)*.

Beberapa contoh editor yang bisa dipakai, yaitu:

- a. *Notepad*
Notepad++ adalah salah satu *enhanced Text Editor* yang sangat populer di *OS Windows*, *lightweight*, dengan dukungan *plug-in* yang banyak sehingga membuat *Notepad++* menjadi aplikasi *Text Editor* yang *powerful*.
- b. *NetBeans*
NetBeans adalah IDE yang bagus. Dalam *NetBeans* terdapat *GUI Builder* yang sudah *built-in* yang dikembangkan dalam Java dan kompatibel untuk berbagai platform. Untuk mengunduh dapat mengunjungi situs resminya di <https://netbeans.org/>.
- c. *Eclipse*
Eclipse adalah IDE yang bagus juga *open source*. *Eclipse* dikembangkan dengan Java yang banyak digunakan untuk kepentingan komersial dan individu, juga kompatibel untuk berbagai *platform*, selain punya banyak dukungan *plugin* tambahan untuk memperluas pengembangan yang dibutuhkan, juga terdapat *highlight* untuk *compiling error*. Untuk mengunduh dapat mengunjungi situs resminya di <https://eclipse.org/>
- d. *JCreator*
JCreator adalah IDE yang bagus dan mudah digunakan. Berbeda dengan sebelumnya, IDE ini dikembangkan dalam C++ bukan dengan Java. *JCreator* hanya tersedia untuk *platform Windows*. Untuk mengunduh dapat mengunjungi situs resminya di <http://www.jcreator.com>
- e. *BlueJ*
BlueJ adalah IDE yang sederhana. Banyak digunakan untuk konsep pemrograman Java dan juga terdapat kanvas UML. Untuk mengunduh dapat mengunjungi situs resminya di <http://www.bluej.org/>

Java terbagi atas 2 bagian utama, yaitu:

- 1) *Java Virtual Machine (JVM)*.
- 2) *Java Application Programming Interface (JavaAPI)*.

SUN *Microsystem* sebagai pengembang *software* Java membagi arsitektur Java menjadi tiga bagian, yaitu:

- 1) *Enterprise Java* (J2EE) untuk aplikasi berbasis web.
- 2) *Standard Java* (J2SE), ini adalah yang biasa dikenal sebagai bahasa Java.
- 3) *Micro Java* (J2ME) merupakan subset dari J2SE dan salah satu aplikasi yang banyak dipakai adalah untuk *wireless device/mobile device*.

Terdapat beberapa fitur yang disediakan oleh Java, yakni:

- 1) **JVM**
JVM adalah sebuah mesin imajiner (maya) yang bekerja dengan menyerupai aplikasi pada sebuah mesin nyata. JVM menyediakan spesifikasi *hardware* dan *platform* perangkat keras yang dapat meng-*compile* semua program Java. Program ini dikompilasi menghasilkan satu berkas *bytecode*.
- 2) *Garbage collection*
Bertanggung jawab untuk mengosongkan memori. Pengosongan memori terjadi otomatis selama masa aktif dari program java. Programmer dibebaskan dari beban untuk mengalokasikan kembali memori itu sendiri.
- 3) *Code security*
Code Security terimplementasi pada Java melalui penggunaan *Java Runtime Environment* (JRE).

Java menggunakan model pengamanan 3 lapis untuk melindungi *system* dari *untrusted Java Code*, yang terdiri atas berikut ini.

- 1) Pertama, *class-loader* menangani pemuatan *class* Java ke *runtime interpreter*.
- 2) Kedua, *byte code verifier* membaca *byte code* sebelum dijalankan dan menjamin *byte code* memenuhi aturan-aturan dasar bahasa Java.
- 3) Ketiga, manajemen keamanan yang menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumberdaya seperti *system file*, *port* jaringan, proses eksternal dan *system windowing*.

Java SDK dan *NetBeans* diperlukan ketika mulai membuat program dengan bahasa pemrograman Java. *Java SDK* adalah *platform* dasar Java yang diperlukan agar komputer atau laptop dapat digunakan untuk mengeksekusi kode-kode program bahasa Java, sedangkan *NetBeans* adalah aplikasi editor terpadu (IDE atau *Integrated Develepment Environment*) yang akan banyak mempermudah dalam membuat aplikasi karena menyediakan kontrol-kontrol visual yang penting dalam pemrograman *desktop* (atau lebih dikenal sebagai pemrograman visual).



Latihan

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Jelaskan mengapa kita memerlukan konsep pemrograman berorientasi objek!

Jawab :

.....
.....

- 2) Sebutkan perbedaan pemrograman berorientasi objek dengan pemrograman *procedural*.

Jawab :

.....
.....

- 3) Jelaskan cara kerja dari *compiler*!

Jawab :

.....
.....

- 4) Jelaskan apa yang dimaksud dengan *Interpreter*!

Jawab :

.....
.....

Petunjuk Jawaban Latihan

Untuk mengerjakan latihan tersebut, silakan pelajari kembali materi pada Kegiatan Belajar 1 ini.



Rangkuman

Istilah “*object oriented programming*” pertama kali digunakan oleh *Xerox PARC* diluar bahasa pemrograman lainnya. Metode pemrograman dalam dunia programming terbagi atas 5 jenis, yaitu (1) Imperatif, (2) Terstruktur, (3) Prosedural, (4) *Object*

Oriented, dan (5) Fungsional. Pada model pemrograman tradisional atau prosedural yang dikenal dengan *process-oriented model*, semua data dan kode digabung menjadi satu bagian dalam satu program. *Object Oriented Programming* (OOP) atau Pemrograman Berorientasi Objek (PBO) adalah metode pemrograman yang berorientasikan kepada objek, yaitu semua data dan *method* didefinisikan ke dalam beberapa *class* atau objek-objek agar bisa saling bekerja sama dalam memecahkan masalah.

Java adalah suatu teknologi di dunia *software* komputer, yang merupakan suatu bahasa pemrograman, dan sekaligus suatu *platform*. Pada pemrograman Java terbagi menjadi dua yaitu *Java Virtual Machine* (JVM) dan *Java Application Programming Interface* (JavaAPI). *Java SDK* dan *NetBeans* diperlukan jika anda hendak mulai membuat program dengan bahasa pemrograman Java. *Java SDK* adalah platform dasar Java yang diperlukan agar komputer atau laptop dapat digunakan untuk mengeksekusi kode-kode program bahasa Java, sedangkan *NetBeans* adalah aplikasi editor terpadu atau *Integrated Develepmnt Environment* (IDE) yang akan banyak mempermudah dalam membuat aplikasi karena menyediakan kontrol-kontrol visual yang penting dalam pemrograman *desktop* (atau lebih dikenal sebagai pemrograman visual).



Tes Formatif 1

Pilihlah satu jawaban yang paling tepat!

- 1) Bahasa pertama yang menggunakan konsep PBO/OOP adalah
 - A. simule
 - B. simula
 - C. simalu
 - D. simale

- 2) Istilah *object oriented programming* (OOP) pertama kali digunakan oleh Xerox PARC di luar bahasa pemrograman lainnya, merujuk pada proses yang menggunakan objek sebagai dasar
 - A. perhitungan
 - B. *class*
 - C. objek
 - D. metode

- 3) Metode pemrograman yang mengeluarkan perintah yang akan dieksekusi oleh komputer. Biasanya berupa baris-baris program yang dieksekusi secara urut mulai dari baris atas hingga bawah disebut dengan
 - A. terstruktur
 - B. prosedural

- C. *object oriented*
D. *text base*
- 4) Model data berorientasi objek dapat memberikan fleksibilitas yang lebih banyak, karena
- A. memberikan kemudahan dalam mengubah program
 - B. digunakan dalam program berskala kecil
 - C. terdiri dari sub-sub modul
 - D. setiap bagian mengandung *class*
- 5) Pada pemrograman berorientasi objek, dimana sebuah objek dapat mempunyai objek turunan disebut
- A. *inheritance*
 - B. *polimorphism*
 - C. *encapsulation*
 - D. *abstraction*
- 6) Setiap *object* memiliki *property*. Berikut yang bukan termasuk *property* dari laptop adalah
- A. merk
 - B. warna
 - C. tipe
 - D. *charger*
- 7) Suatu program yang menterjemahkan bahasa program ke dalam bahasa objek adalah
- A. *interpreter*
 - B. editor
 - C. *compiler*
 - D. translator
- 8) Untuk menulis *source code* maka kita memerlukan
- A. aplikasi
 - B. *linker*
 - C. editor
 - D. *assembly*

- 9) Ketika kita melakukan proses *compile* suatu kode program, posisi *Loader* setelah
- A. editor
 - B. *linker*
 - C. CPU
 - D. *compiler*
- 10) Perangkat lunak yang mampu mengeksekusi kode program (yang ditulis oleh *programmer*) lalu menterjemahkannya ke dalam bahasa mesin, sehingga mesin melakukan instruksi yang diminta oleh *programmer* tersebut
- A. *translator*
 - B. *compiler*
 - C. *interpreter*
 - D. *assembly*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat Penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100$$

Arti tingkat penguasaan



Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

Instalasi Program Java

Pada Kegiatan Belajar 2 ini, masing-masing mahasiswa diharapkan untuk mempersiapkan komputer dengan *software* Java dan *Eclipse* sebagai editornya, agar proses pembelajaran dapat berjalan dengan optimal. Untuk instalasi *software* Java ini, digunakan versi Java terbaru yaitu JDK 12. Apabila versi Java yang diinstal berbeda, cara instalasinya hampir sama.

Proses *install* Java di Windows terdiri dari dua bagian utama, yaitu instalasi Java dan pengaturan (*setting*) variabel-variabel sistem (*system variables*) dan *path* pada Windows.

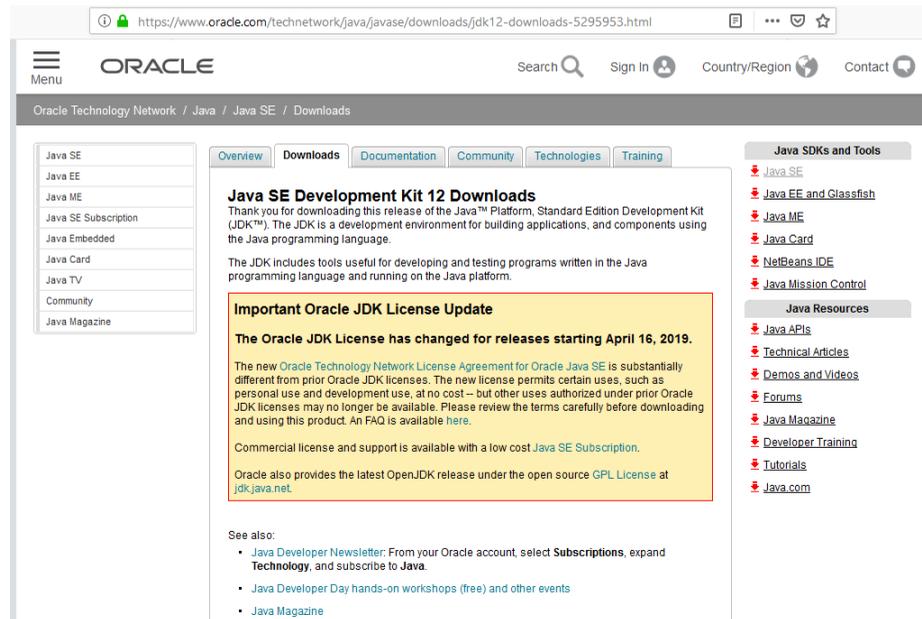
Persyaratan-persyaratan sebelum *install* Java adalah harus sudah tersedia file penginstal java (*Java installer*). Jika belum memiliki file tersebut, silakan kunjungi terlebih dahulu di *website* resminya dan *download* file *Java installer*. Setelah Anda kunjungi *link* tersebut maka akan muncul tampilan seperti gambar di bawah

A. LANGKAH INSTALASI JAVA



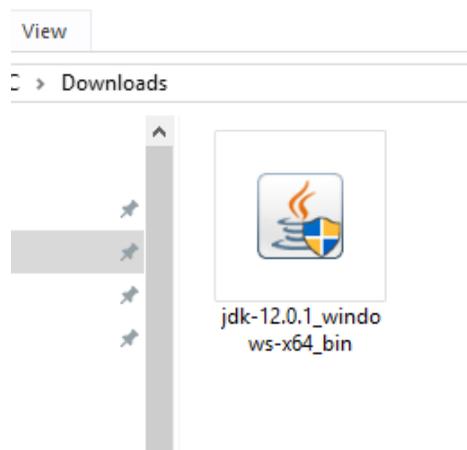
Gambar 1.5
Halaman Utama Web *Download* Java

Untuk bisa *download* file instalasi Anda harus setuju dengan *license agreement Java*, dengan cara klik *radio button* yang memiliki tulisan **Accept License Agreement**. Selanjutnya silakan *download file* instalasi Java sesuai OS Anda. Jika untuk Windows 64 bit, maka silakan memilih file yang bernama **jdk-12.0.1_windows-x64_bin**.



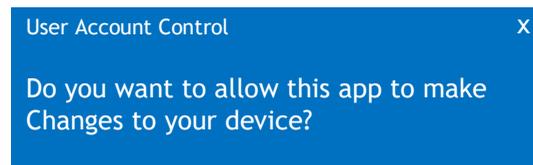
Gambar 1.6
Persetujuan untuk *Install* Java

Setelah file Java selesai di *download* maka untuk menginstalnya cukup dengan klik 2 kali pada *file* Java hasil *download* tersebut.



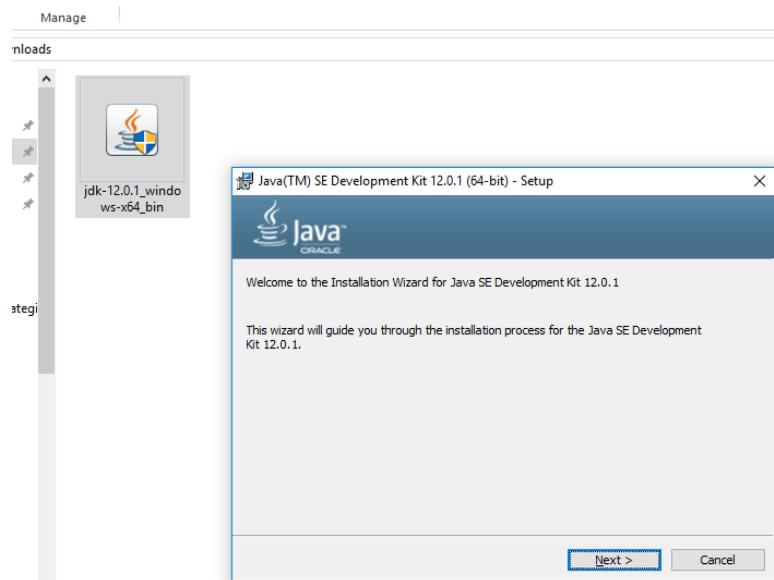
Gambar 1.7
File Java yang Sudah di Download

Jika muncul popup UAC (*User Account Control*) seperti Gambar 1.8 dibawah silakan klik **Yes**.



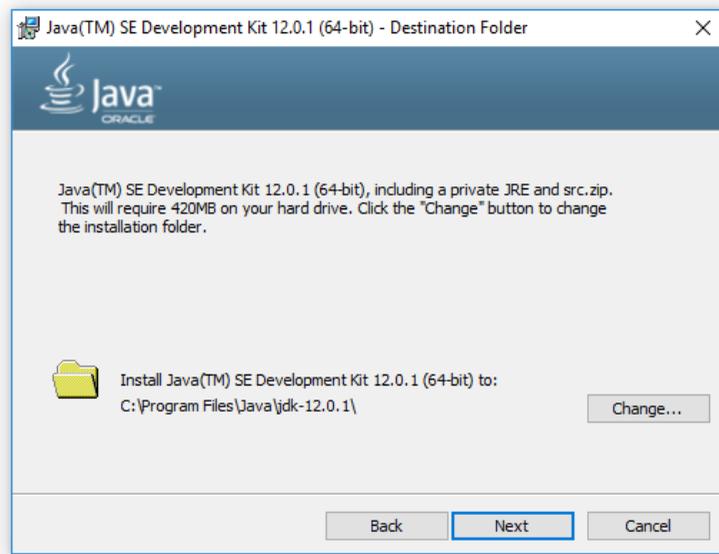
Gambar 1.8
Persetujuan Memasang Java di Komputer

Kemudian akan tampil Java SE setup seperti Gambar 1.9 di bawah, pada langkah ini klik **Next**.



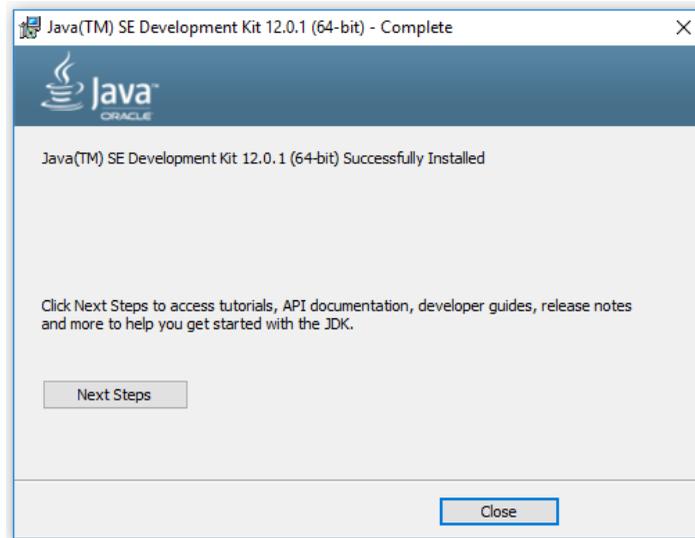
Gambar 1.9
Setup Program Java

Pada langkah *Custom Setup* seperti Gambar 1.10 di bawah, biarkan *default* saja, lalu klik **Next**.



Gambar 1.10
Proses *Install* Java

Apabila muncul *Custom Setup* lagi seperti Gambar 1.11 di bawah, klik **Next**.



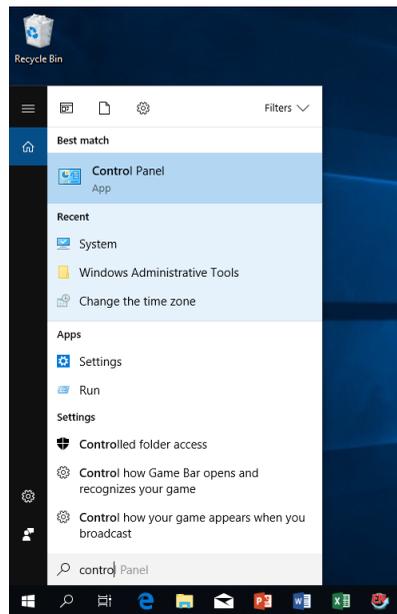
Gambar 1.11
Proses *Install* Java 2

Setelah muncul tampilan seperti di Gambar 1.11 maka Anda telah berhasil meng-*install* Java. Pada langkah ini silakan klik *Close*.

Sampai di sini Anda telah berhasil *install* Java di PC Windows Anda, namun ini belum selesai. Kita masih perlu menambahkan *system* variabel dan *path* pada windows.

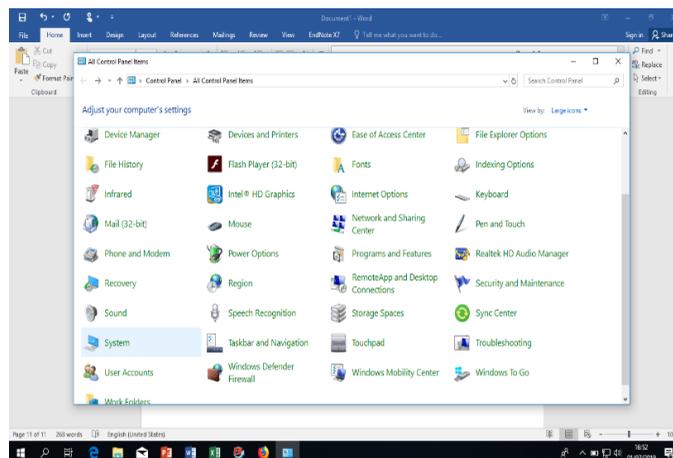
B. *SETTING SYSTEM VARIABLE DAN PATH JAVA PADA WINDOWS*

Langkah pertama untuk *setting system variable* dan *path* Java pada Windows 10 silakan buka menu *control panel* di PC Anda. Dengan cara klik *Start* (logo *Windows*) dan tulis *control panel*, maka akan muncul hasil pencarian seperti Gambar 1.12. Kemudian silakan pilih *Control Panel*.



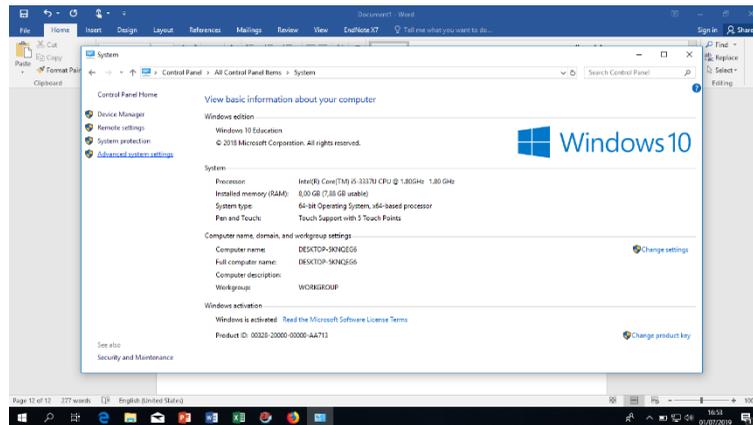
Gambar 1.12
Setting System Variable dan Path

Setelah menu *Control Panel* dibuka seperti pada Gambar 1.13, maka selanjutnya pilih *System*.



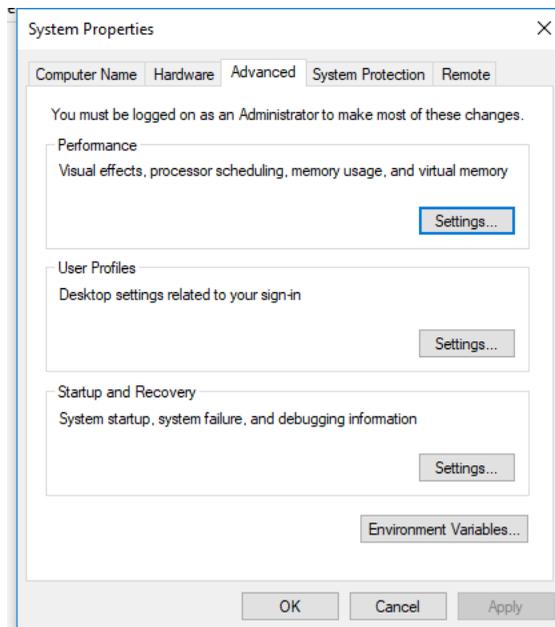
Gambar 1.13
Menu Control Panel

Setelah menu *System* dibuka Gambar 1.14, maka selanjutnya pilih *Advanced system settings*.



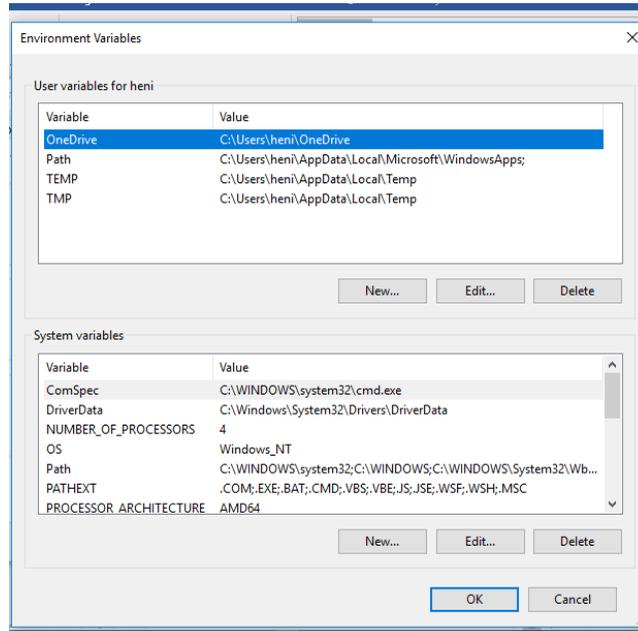
Gambar 1.14
Menu *System*

Maka akan muncul tampilan seperti Gambar 1.15, lalu klik **Environment Variables**.



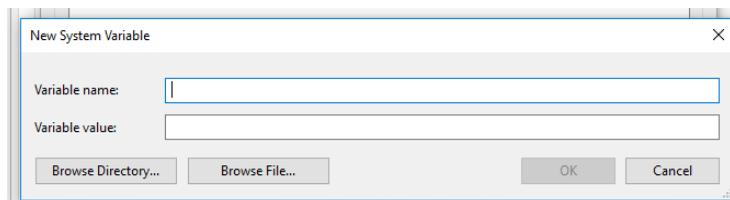
Gambar 1.15
System Properties

Pada langkah selanjutnya, seperti Gambar 1.16 silahkan alihkan perhatian Anda ke *frame* bernama *System variables*. Pada *frame* tersebut dan klik tombol **New**.



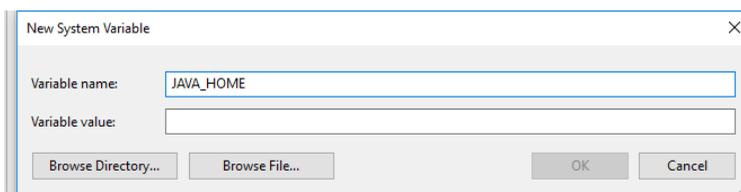
Gambar 1.16
Menu *Environment Variables*

Maka akan muncul tampilan *New System Variable* seperti Gambar 1.16 berikut



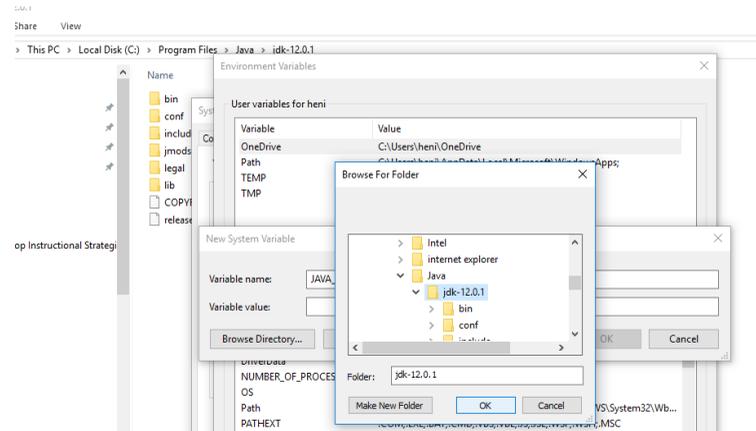
Gambar 1.17
Menu *System Variable*

Silakan isi **Variable name** dengan nama JAVA_HOME seperti pada Gambar 2.17. Selanjutnya klik **Browse Directory....**



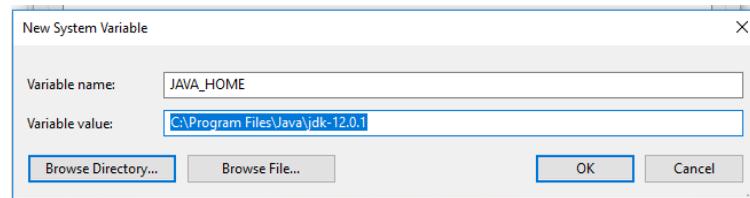
Gambar 1.18
Gambar Menu *System* Setelah Diisi

Selanjutnya, cari *path folder* instalasi Java sesuai Gambar 1.19, dan pilih *folder* `jdk-12.0.1`. Pada contoh ini *path* jdk Java saya berada di *directory* “`C:\Program Files\Java\jdk-12.0.1`”. Kemudian klik **OK**



Gambar 1.19
Lokasi Program Java pada Komputer

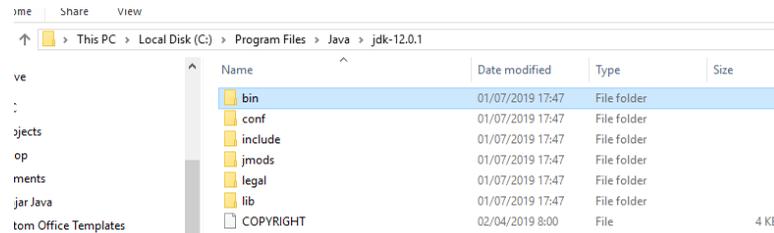
Setelah itu **Variabel value** pada Gambar 1.20 akan terisi *path* jdk Java Anda, lanjutkan dengan klik **OK**



Gambar 1.20
Tampilan *New System Variabel*

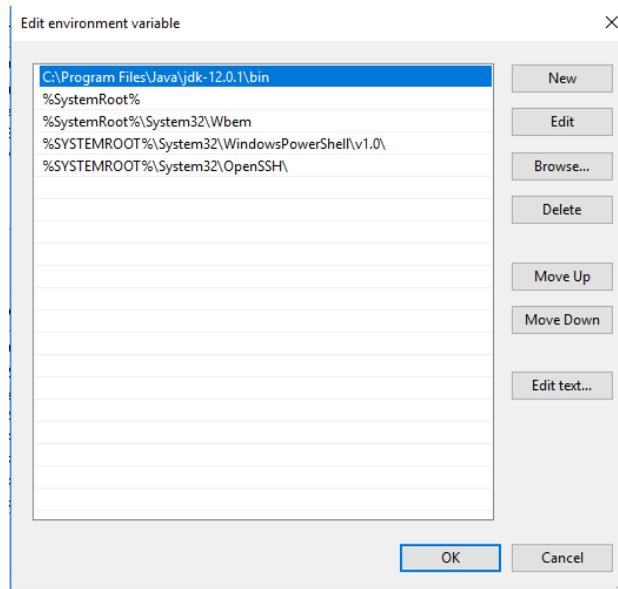
Langkah selanjutnya pilih *Path* dan klik **Edit** (lihat Gambar 1.21).

Klik menu **Browse** pada Gambar 1.22, maka akan tampil Gambar 1.23



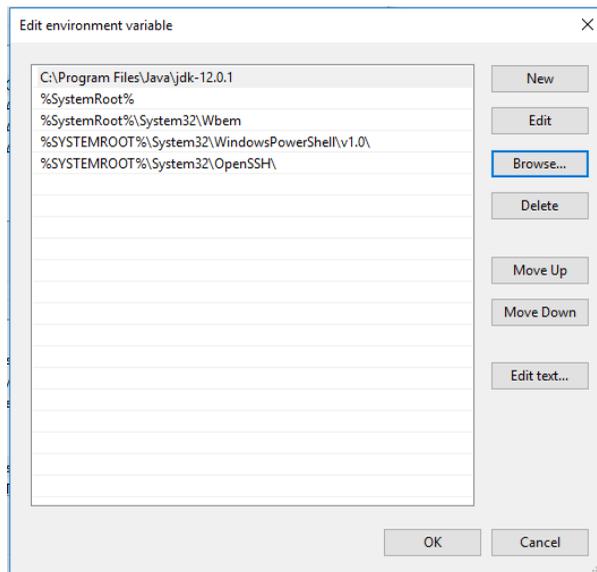
Gambar 1.23
File JDK

Setelah itu buka *File Explorer*, dan cari *folder bin* pada instalasi Java. Biasanya terletak di dalam *directory C:\Program Files\Java*. Tampilan Gambar 1.24 memperlihatkan hasil dari isi *folder bin* berikut.



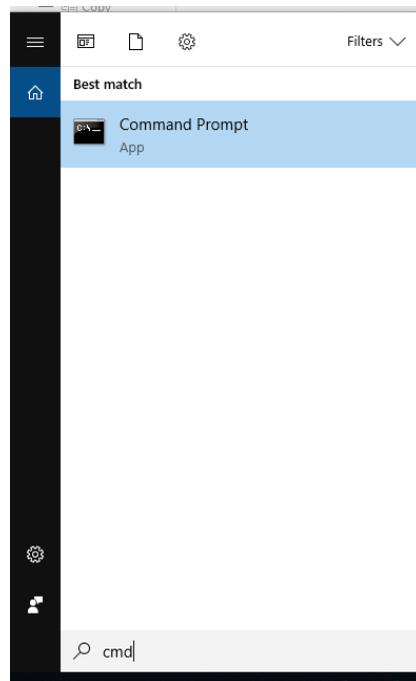
Gambar 1.24
Edit *Environment Variable*

Klik **Browse**, seperti pada Gambar 1.25



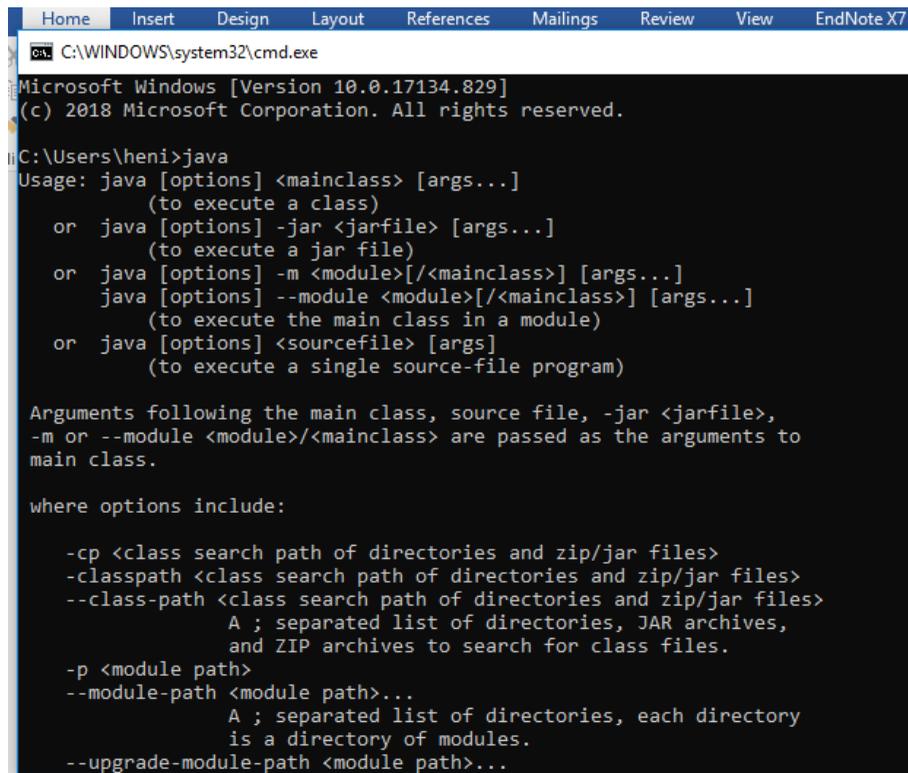
Gambar 1.25
Edit Environment Variable

Ketik *cmd* pada *menu search* Gambar 1.26, kemudian *double klik Command Prompt*



Gambar 1.26
Lokasi Command Prompt

Pada *prompt* ketikkan tulisan Java kemudian tekan tombol **enter**, maka akan tampil path Java seperti Gambar 1.27 berikut ini.



```
Home Insert Design Layout References Mailings Review View EndNote X7
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\heni>java
Usage: java [options] <mainclass> [args...]
        (to execute a class)
    or java [options] -jar <jarfile> [args...]
        (to execute a jar file)
    or java [options] -m <module>[/<mainclass>] [args...]
        java [options] --module <module>[/<mainclass>] [args...]
        (to execute the main class in a module)
    or java [options] <sourcefile> [args]
        (to execute a single source-file program)

Arguments following the main class, source file, -jar <jarfile>,
-m or --module <module>/<mainclass> are passed as the arguments to
main class.

where options include:

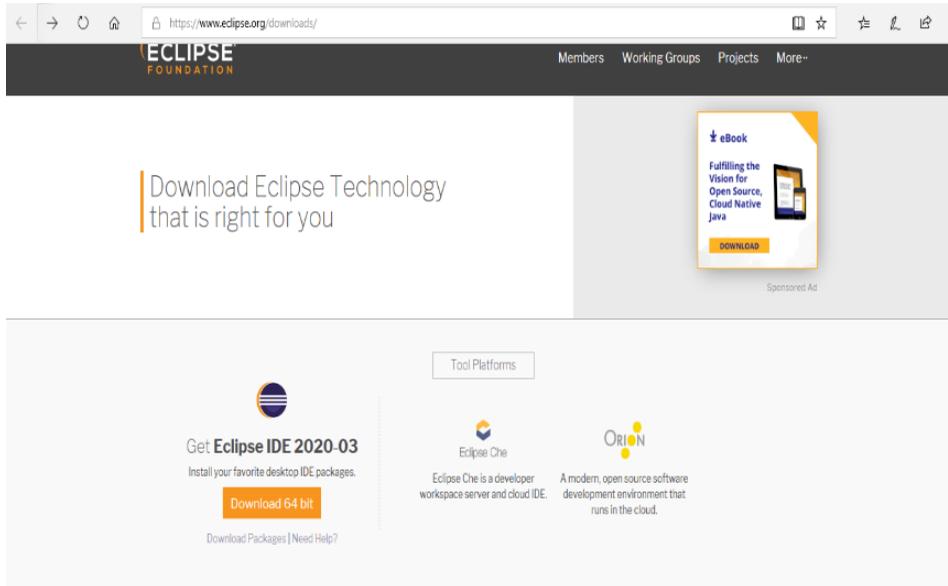
    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
    --class-path <class search path of directories and zip/jar files>
                A ; separated list of directories, JAR archives,
                and ZIP archives to search for class files.
    -p <module path>
    --module-path <module path>...
                A ; separated list of directories, each directory
                is a directory of modules.
    --upgrade-module-path <module path>...
```

Gambar 1.27
Tampilan *Command Prompt*

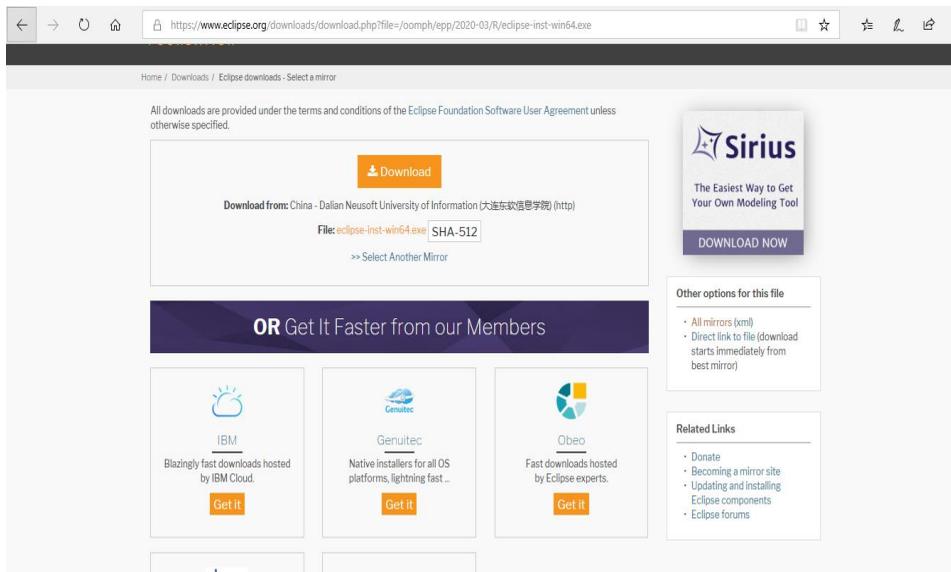
Demikian tata cara instalasi Java di Windows 10 64 bit.

- **Instalasi Eclipse**, untuk menginstal *Eclipse* terlebih dahulu *download* IDE di link berikut <https://www.eclipse.org/downloads/>.

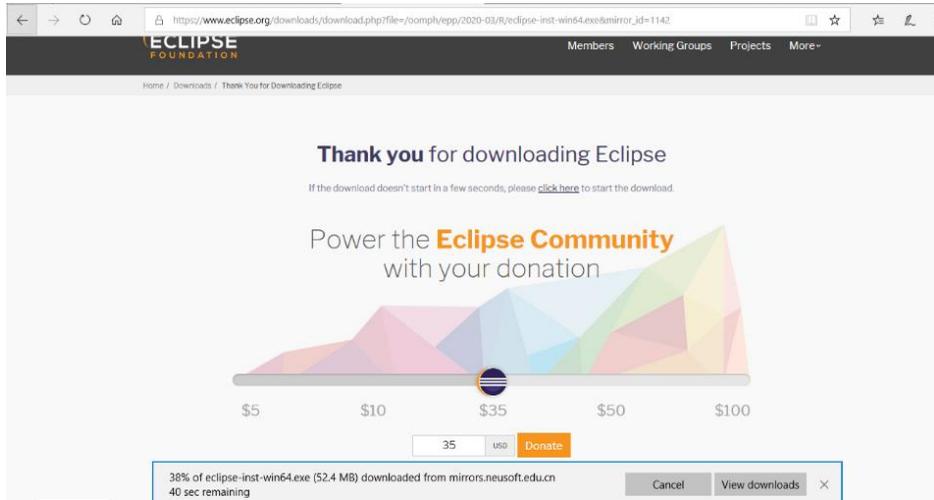
Selanjutnya akan ditampilkan langkah-langkah instalasi *Eclipse* sampai dengan tampilan lembar kerja.



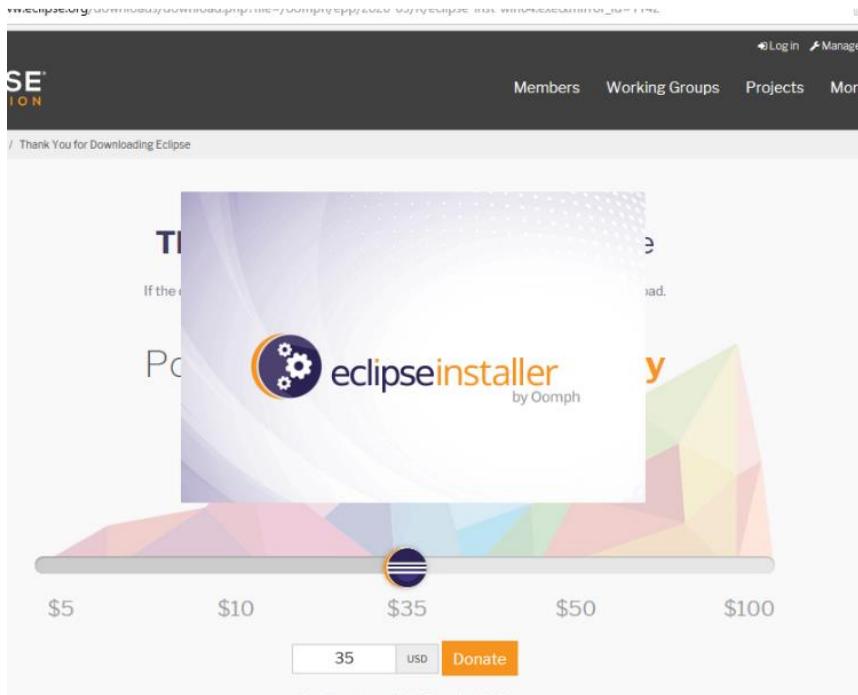
Gambar 1.28
Tampilan Web *Install Eclipse*



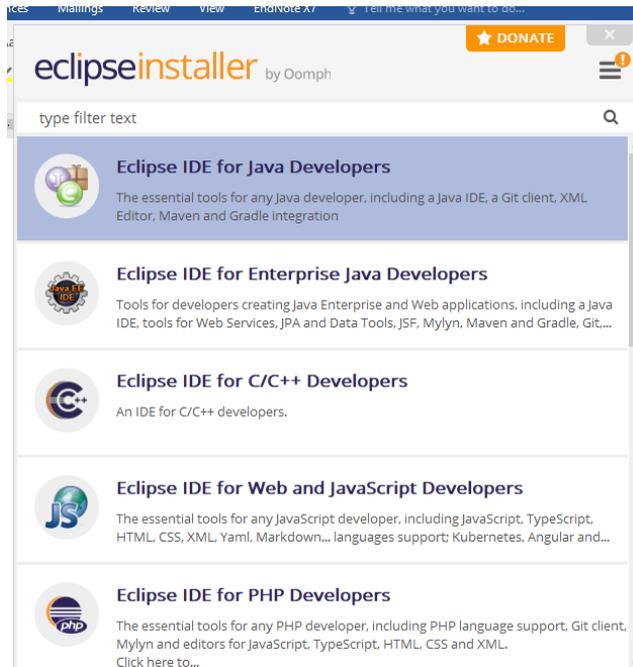
Gambar 1.29
Menu *Install Eclipse*



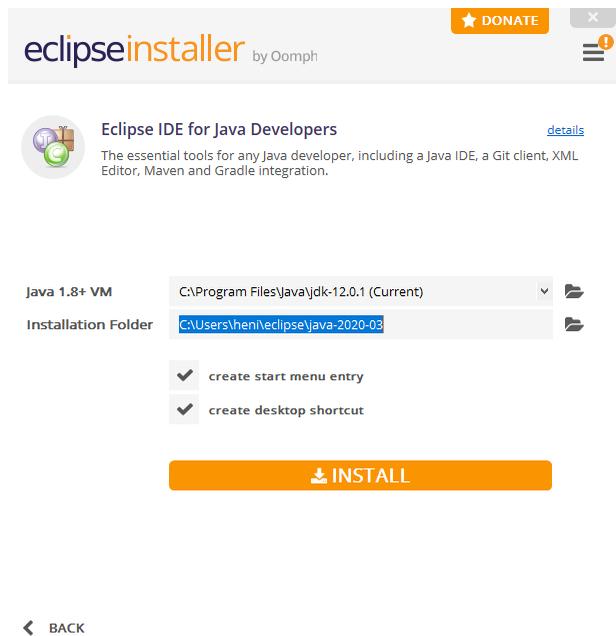
Gambar 1.30
Proses *Install*



Gambar 1.31
Tampilan *Install* Eclipse

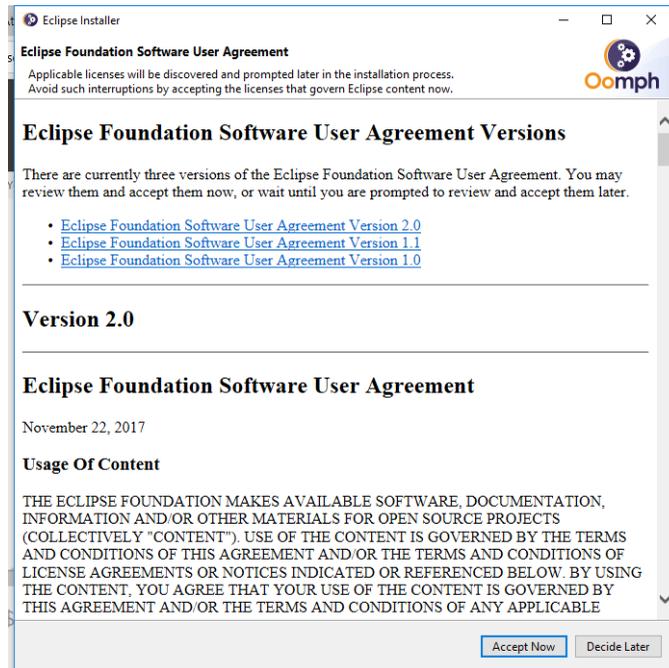


Gambar 1.32
Pilihan Eclipse *Installer*

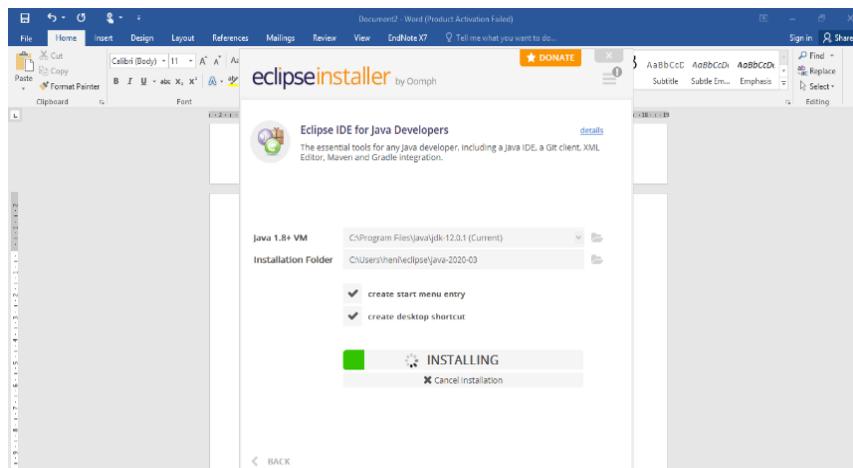


Gambar 1.33
Instalasi

Setelah klik **INSTALL**, akan muncul tampilan untuk *User Agreement* seperti pada Gambar 1.34. Selanjutnya pilih **Accept Now**.



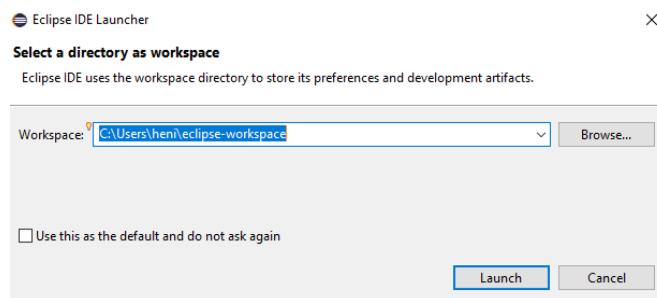
Gambar 1.34
Tampilan *User Agreement*



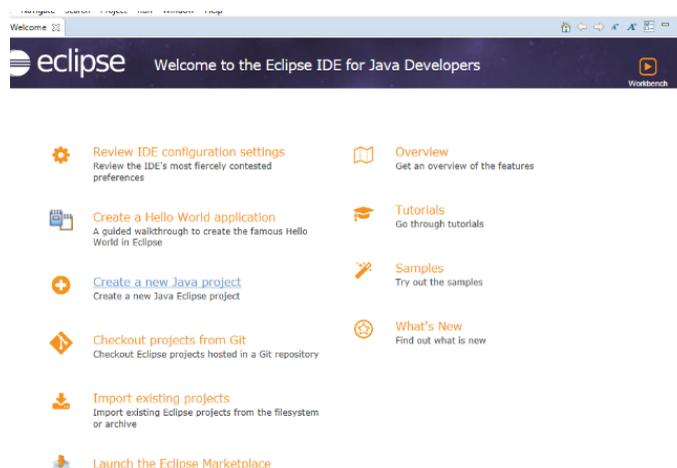
Gambar 1.35
Install IDE for Java



Gambar 1.36
Tampilan Setelah *Install* Selesai

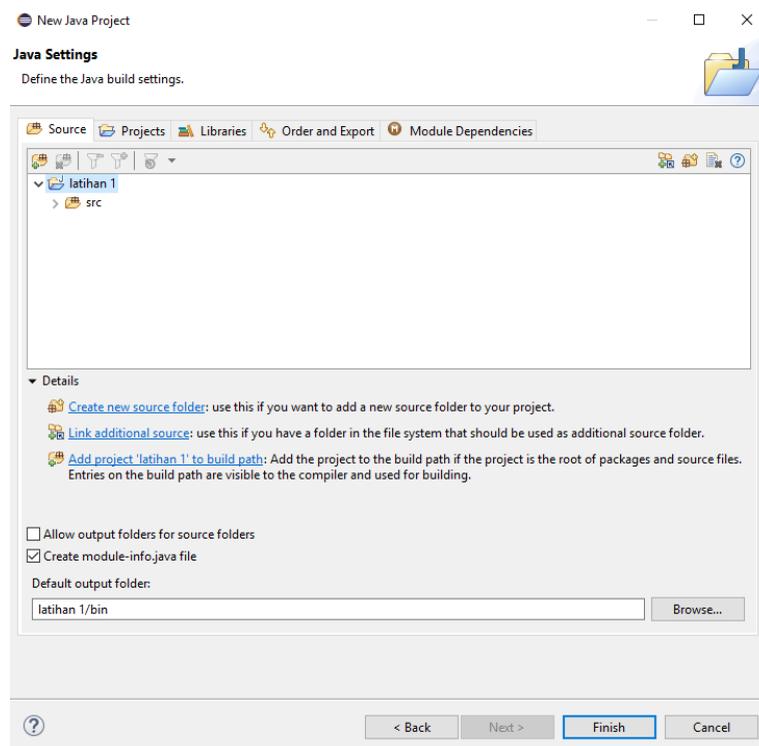
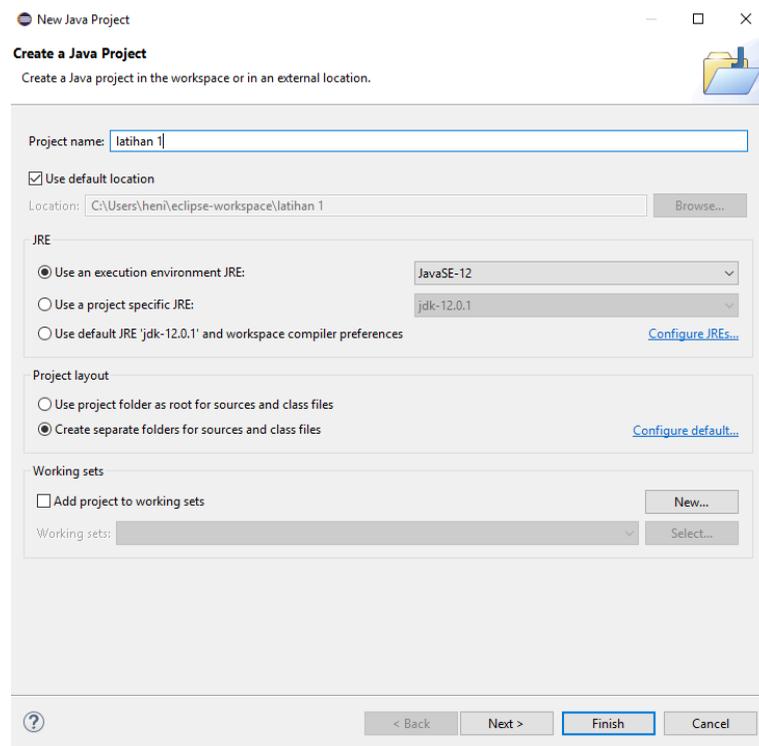


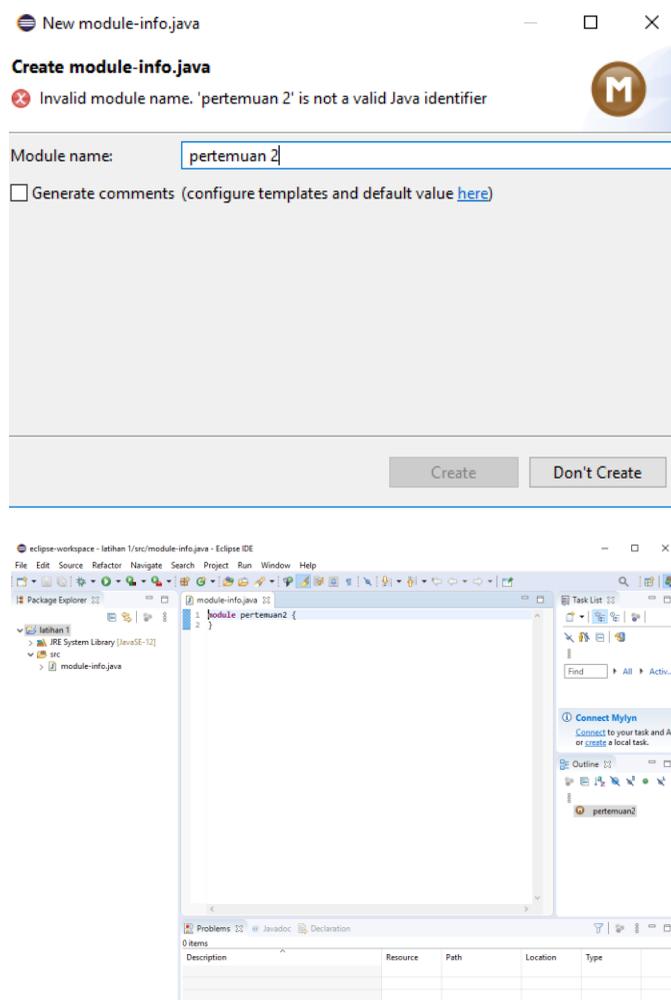
Gambar 1.37
Tampilan Pemilihan Lokasi Aplikasi *Eclipse* Hasil Instalasi



Gambar 1.38
Tampilan Pilihan Aplikasi *Eclipse*

Jika kita pilih **Create a new Java Project** pada tampilan Gambar 1.38, maka akan muncul tampilan-tampilan berikut.





Gambar 1.39
Lembar Kerja *Eclipse*



Latihan

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- Lakukan instalasi *software* Java beserta editornya sebelum kita memasuki Modul 2!

Petunjuk Jawaban Latihan

Untuk mengerjakan latihan tersebut, ikuti petunjuk dan langkah-langkah yang telah disampaikan pada materi Kegiatan Belajar 2.



Rangkuman

Salah satu faktor yang mengakibatkan tidak lancarnya proses *install* Java adalah spesifikasi komputer yang tidak sesuai, karena untuk menjalankan program Java dibutuhkan spesifikasi komputer yang cukup tinggi.

Java Development Kit atau biasa di sebut JDK adalah *software* yang digunakan untuk melakukan proses kompilasi. Langkah pertama *install* adalah *men-download* JDK dari situs resminya.



Tes Formatif 2

Pilihlah satu jawaban yang paling tepat!

- 1) JDK merupakan singkatan dari
 - A. *Java Desain Kit*
 - B. *Java Design Kit*
 - C. *Java Developer Kit*
 - D. *Java Development Kit*

- 2) Saat melakukan *install* Java, ada beberapa point yang harus dilakukan, *kecuali*
 - A. *Install Java*
 - B. *Setting system*
 - C. *Setting computer*
 - D. *Setting path*

- 3) *Jdk-12.0.1_windows-x64_bin* artinya adalah
 - A. Java versi 12
 - B. Java versi 12 untuk Windows
 - C. Java versi 12 untuk Windows 64
 - D. Java versi 12 untuk Windows bin

- 4) Langkah untuk *install* Java
 - A. download file java – accept license agreement – double klik file hasil download – ikuti klik next-close
 - B. download file java-double klik file hasil download – accept license agreement - ikuti langkah dengan klik next-close

- C. download file java – ikuti langkah dengan klik next-accept license agreement – close
- D. download file java – double klik file hasil download – ikuti langkah dengan klik next-close
- 5) Langkah untuk *setting* System Variabel
- A. klik control panel – system – advance setting-environment variabel–new–isi data – browse directory – cari lokasi file jdk – OK
- B. klik control panel – advance setting- environment variabel- new- system
- C. klik control panel – new–system–advance setting–environment variabel–new–isi data- browse directory–cari lokasi file jdk
- D. klik control panel – browse directory – advance setting– nvironment variabel–new – isi data – browse directory – cari lokasi file jdk
- 6) Berikut ini adalah editor yang dapat digunakan untuk menuliskan program Java, *kecuali*
- A. *JCreator*
- B. *Notepad*
- C. *Eclipse*
- D. *JavaBeans*
- 7) Sintaks java untuk melakukan kompilasi terhadap berkas program adalah
- A. Java
- B. Javac
- C. Javaclass
- D. Javax
- 8) Hasil kompilasi dari berkas Java adalah
- A. file BAK
- B. *file Bytecode*
- C. *file executable*
- D. *file class*
- 9) Editor yang berbasis *text* adalah
- A. *JCreator*
- B. *Notepad*
- C. *Eclipse*
- D. *JavaBeans*

- 10) Arsitektur Java yang digunakan untuk membuat aplikasi berbasis web adalah
- A. J2EE
 - B. J2SE
 - C. J2ME
 - D. JVM

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat Penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100$$

Arti tingkat penguasaan



Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

Kunci Jawaban Tes Formatif

Tes Formatif 1

- 1) B
- 2) A
- 3) B
- 4) A
- 5) A
- 6) D
- 7) C
- 8) C
- 9) A
- 10) C

Tes Formatif 2

- 1) D
- 2) C
- 3) C
- 4) A
- 5) A
- 6) D
- 7) B
- 8) B
- 9) A
- 10) B

| | | |
|---------------------------------|---|--|
| <i>Compiler</i> | : | Suatu program yang menterjemahkan bahasa program (<i>source code</i>) kedalam bahasa objek. |
| <i>Install</i> | : | Memasang program (perangkat lunak) ke dalam komputer. |
| <i>Interpreter</i> | : | Perangkat lunak yang mampu mengeksekusi kode program (yang ditulis oleh <i>programmer</i>) lalu menterjemahkannya ke dalam bahasa mesin, sehingga mesin melakukan instruksi yang diminta oleh <i>programmer</i> tersebut. |
| Java SDK | : | <i>Platform</i> dasar java yang diperlukan agar komputer atau laptop dapat digunakan untuk mengeksekusi kode-kode program bahasa java. |
| JVM | : | Singkatan dari <i>Java Virtual Machine</i> . |
| <i>Linker</i> | : | Suatu program yang menterjemahkan program objek (berekstensi OBJ) ke bentuk program eksekusi. |
| Objek | : | Entitas dasar <i>run-time</i> dalam suatu sistem berorientasi objek. |
| OOP/PBO | : | Singkatan dari <i>Object Oriented Programming</i> / Pemrograman Berorientasi Objek. |
| Pemrograman Berorientasi Objek: | | Metode pemrograman yang berorientasikan kepada objek. |
| Pemrograman Prosedural | : | Metode pemrograman yang mengeluarkan perintah yang akan dieksekusi oleh komputer. |
| <i>Uninstal</i> | : | Melepaskan hasil instalasi program yang ada. |

Daftar Pustaka

Hosrtmann, C. (2015). *Big Java early objects*. USA: Wiley.

<http://objekaja.blogspot.com/2008/12/sejarah-pemrograman-berorientasi-objek.html>

<https://mybiodatakarina.wordpress.com/2015/08/06/perbedaan-pemograman-berorientasi-objek-pbo-dengan-pemograman-prosedural/>

<http://www.infomugi.com/2013/04/pengertian-compiler-interpreter.html>

<https://www.codepolitan.com/ide-terbaik-untuk-java>