1. Bahasa pemrograman Java terlahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, dan James Gosling, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot *Duke* yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java Oak* pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan TV kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan peramban (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "Bapak Java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak lain sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java).

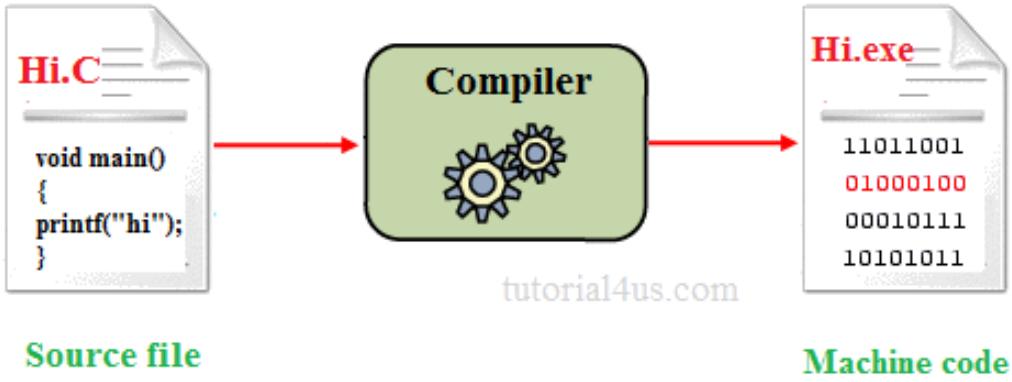
2. (a). Compiler adalah suatu program yang menerjemahkan bahasa program (source code) kedalam bahasa objek (obyek code). Compiler menggabungkan keseluruhan bahasa program, mengumpulkannya dan kemudian menyusunnya kembali.

Komplier memerlukan waktu untuk membuat suatu program dapat di eksekusi oleh computer, program yang dieksekusi oleh compiler adalah dapat berjalan lebih cepat disbanding program yang diperoduksi oleh interpreter, disamping itu juga bersifat independen. Contoh program yang menggunakan compiler adalah Visual Basic, Visual Delvi, dan Pascal.

- Tahap Kompilasi:

1. Pertama source code (program yang ditulis) dibaca kememori computer).
2. Source code tersebut diubah menjadi objek code (bahasa Assembly).
3. Objek code di hubungkan dengan liberary yang dibutuhkan untuk membentuk file yang bisa dieksekusi.

Cara Kerja Interpreter



b. 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 tersebut. Perintah-perintah yang dibuat oleh programmer tersebut dieksekusi baris demi baris, sambil mengikuti logika yang terdapat di dalam kode tersebut.

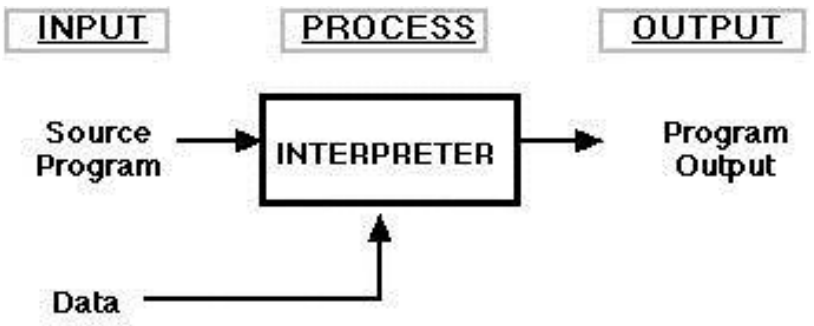
Proses ini sangat berbeda dengan compiler, dimana pada compiler, hasilnya sudah langsung berupa satu kesatuan perintah dalam bentuk bahasa mesin, dimana proses penterjemahan dilaksanakan sebelum program tersebut dieksekusi.

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.

- Adapun fungsi utama dari penerjemah dalam computing ialah :

1. Mengeksekusi kode sumber secara langsung, atau
2. Menerjemahkannya ke dalam serangkaian p-code kemudian mengeksekusinya, atau
3. Mengeksekusi kode yang telah dikompilasi sebelumnya oleh kompileryang merupakan bagian dari sistem penerjemahan.

Cara Kerja Interpreter



3.Pemrograman berorientasi objek (Inggris: ***object-oriented programming*** disingkat **OOP**) merupakan paradigma pemrograman berdasarkan konsep "objek", yang dapat berisi data, dalam bentuk *field* atau dikenal juga sebagai atribut; serta kode, dalam bentuk fungsi/prosedur atau dikenal juga sebagai *method*. Semua data dan fungsi di dalam paradigma ini dibungkus dalam *kelas-kelas* atau *objek-objek*. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya,

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik peranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Konsep dasar

- Kelas — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. *Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada,* dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
- Objek - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
- Abstraksi - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
- Enkapsulasi - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
- Polimorfisme melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.
- Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

4. **Java Deathmatch, Uji Skill Pemrograman Javamu**

```
public class HelloVariable {

    public static void main(String[] args) {
        int umur = 10;
        double gravitasi = 9.8;
        String nama = "Peter Parker";
        Boolean bukan_dosen = false;
        String[] superhero = new String[]{"Ant Man", "Captain America",
"Spiderman", "Hulk", "Thor", "Iron Man"};

        System.out.println("umur: " + umur);

        System.out.println("gravitasi: " + gravitasi);

        System.out.println(nama instanceof String);
        System.out.println("nama: " + nama);

        System.out.println(superhero instanceof String[]);
        System.out.println("superhero 1: " + superhero[0]);
        System.out.println("superhero 2: " + superhero[1]);
        System.out.println("superhero 3: " + superhero[2]);
        System.out.println("superhero 4: " + superhero[3]);
        System.out.println("superhero 5: " + superhero[4]);
```

```
          System.out.println("superhero 6: " + superhero[5]);
      }

  }
```