1.

a. Class dan Objek

Objek merupakan instance dari class. Jika class secara umum mepresentasikan (template) sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri. Contoh : Dari class Fruit kita dapat membuat objek seperti apel, pisang, jeruk, dan lain-lain.

Class merupakan suatu blueprint atau cetakan untuk menciptakan suatu instant dari object. class juga merupakan grup suatu object dengan kemiripan attributes/properties, behaviour dan relasi ke object lain. Contoh : Class Person, Vehicle, Tree, Fruit dan lain-lain.

b. State (instance variables) –> variable2 yang dideklarasikan di dalam class. Pada contoh di atas nama dan umur disebut dengan state.

Behaviour (methods) –> fungsi2 yang dideklarasikan di dalam class. Pada contoh di atas getNama(), setNama(), getUmur(), dan setUmur() adalah suatu behaviour/method.

2. Overloading dan overriding

Overloading adalah deklarasi method maupun konstruktor lebih dari satu kali pada kelas yang sama. Terdapat beberapa syarat overloading yaitu memiliki jumlah parameter berbeda dan jika jumlah parameter sama tipe data harus berbeda. Method overloading bisa saja terjadi dalam satu class yang sama, dan bisa juga dari class turunan.

Contoh program:

**class** Komputer {

  String cekInfo(**int** memory) {

**return** "Komputer dengan "+memory+" GB RAM";

  }

}

**class** Laptop **extends** Komputer {

  String cekInfo(String pemilik) {

**return** "Ini Laptop milik " + pemilik;

  }

}

**class** BelajarJava {

**public** **static** **void** main(String args[]){

    Laptop laptopAndi = **new** Laptop();

    System.out.println(laptopAndi.cekInfo(16));

    System.out.println(laptopAndi.cekInfo("Andi"));

  }

}

Sedangkan, method overriding adalah pendeklarasian ulang sebuah method pada subclass-nya. Berbeda dengan overloading yang mana method dideklarasikan lebih dari satu kali pada kelas yang sama, maka pada overriding method dideklarasikan kembali pada subclass-nya. Method yang meng-override disebut override method, sedangkan method yang di-override disebut sebagai overriden method.

Contoh program:

//Superclass/Class Induk

class hewan{

//Variable (Default)

String jenis = "Kucing";

int umur = 3;

//Method Get (Default)

String getJenis(){

return jenis;

}

int getUmur(){

return umur;

}

}

//Subclass/Kelas Anak

public class karnivora extends hewan{

public static void main(String[] args){

karnivora data = new karnivora();

}

}

public class karnivora extends hewan{

String jenis = "Anjing";

int umur = 5;

public static void main(String[] args){

karnivora data = new karnivora();

System.out.println("Jenis Hewan: "+data.getJenis());

System.out.println("Usia Hewan: "+data.getUmur());

}

@Override

String getJenis(){

return jenis;

}

@Override

int getUmur(){

return umur;

}

}

3. Keuntungan yang diperoleh dengan melakukan pewarisan adalah

Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam superclass, sifat ini secara otomatis diwariskan dari semua subclasses. Jadi, Kita dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass. Subclass hanya perlu mengimplementasikannya perbedaan sendiri dan induknya.

Keuntungan penggunaan konsep pewarisan antara lain:

* Kita dapat menggunakan kembali kelas-kelas yang kita buat (sebagai superkelas) dan membuat kelas-kelas baru berdasar superkelas tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki superkelas.
* Kita dapat membuat superkelas yang hanya mendefinisikan behaviour namun tidak memberi implementasi dari metode-metode yang ada. Hal ini berguna jika kita ingin membuat semacam template kelas. Kelas semacam ini disebut kelas abstrak, karena behaviournya masih abstrak dan belum diimplementasikan. Subkelas-subkelas dari kelas semacam ini, yang disebut kelas konkret, mengimplementasikan behaviour abstrak tersebut sesuai dengan kebutuhan masing-masing.
* Subclass menyediakan state/behaviour yang spesifik yang membedakannya dengan superclass, hal ini akan memungkinkan programmer Java untuk menggunakan ulang source code dari superclass yang telah ada.
* Programmer Java dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut abstract class, untuk mendefinisikan class dengan behaviour dan state secara umum.

4.