PRAKTIK PEMROGRAMAN BERORIENTASI OBJEK

UJIAN AKHIR SEMESTER 2

Disusun oleh :

NAMA : Clariva Meydieta Widagdo

NIM : V3922012

PS D-III TEKNIK INFORMATIKA

SEKOLAH VOKASI

UNIVERSITAS SEBELAS MARET

2023

1. Kapan memanfaatkan metode pemrograman berorientasi object?
   Jawab :
   Metode pemrograman berorientasi objek berguna ketika ingin mengorganisir ataupun mengelompokkan kode menjadi entitas yang terpisah dan terstruktur. Hal ini berguna ketika program semakin kompleks dengan tujuan agar lebih mudah untuk dipahami. Baik dalam bentuk pengelompokan class(modularitas), konsep pewarisan, enkapsulasi, modularitas, dan lain-lain.

2. Apa manfaat dari penggunaan metode pemrograman berorientasi object?
   Jawab :
   Penggunaan metode pemrograman berorientasi objek sangat penting dan bermanfaat guna memecahkan kompleksitas yang terjadi dan memungkinkan script dapat berinteraksi dari satu kelas ke kelas yang lain. Metode pemrograman berorientasi objek memiliki banyak manfaat yang diantaranya :

   1. Kelas (Class): Kelas berguna sebagai pokok untuk menciptakan objek. Kelas mendefinisikan atribut (data) dan metode (fungsi) yang akan dimiliki oleh objek yang dibuat berdasarkan kelas tersebut.
   2. Objek (Object): Objek menjadi instance konkret dari sebuah kelas.
   3. Pewarisan (Inheritance): Pewarisan memungkinkan kelas untuk mewarisi atribut dan metode dari kelas lain yang disebut kelas induk atau superclass.
   4. Polimorfisme (Polymorphism): Polimorfisme mengacu pada kemampuan objek untuk mengambil banyak bentuk.
   5. Enkapsulasi (Encapsulation): Enkapsulasi berguna untuk detail implementasi internal suatu objek dan hanya mengekspos antarmuka publik yang diperlukan.
   6. Abstraksi (Abstraction): Abstraksi memungkinkan untuk membuat kelas abstrak yang tidak dapat diinstansiasi langsung, tetapi menyediakan kerangka kerja untuk kelas-kelas turunannya.
   7. Paket (Package): Paket digunakan untuk mengorganisir dan mengelompokkan kelas-kelas terkait ke dalam unit yang lebih besar dan membantu dalam mengelola kompleksitas dan mencegah konflik nama kelas yang sama.
   8. Access Modifier: Access modifier digunakan untuk mengatur tingkat aksesibilitas atribut dan metode dalam kelas.
   9. Konstruktor (Constructor): Konstruktor digunakan untuk membuat objek baru dari kelas tersebut.
   10. Getter dan Setter: Getter dan setter adalah untuk mengakses (getter) dan mengubah (setter) nilai atribut dalam sebuah objek.
   11. Exception Handling: Exception handling digunakan untuk menangani situasi yang tidak diharapkan atau kesalahan dalam program.
   12. Inner Class: Inner class dapat mengakses semua atribut dan metode dari kelas luar, bahkan yang bersifat private.
   13. Exception: Exception untuk menunjukkan adanya kesalahan atau situasi yang tidak diharapkan dalam program. Dengan menggunakan exception handling, kita dapat menangani dan memberikan respons yang tepat terhadap exception tersebut.
   14. Error: Error merupakan kondisi yang menunjukkan adanya kesalahan serius yang biasanya tidak dapat diperbaiki dan dapat menyebabkan program berhenti. Contoh error termasuk OutOfMemoryError dan StackOverflowError.
   15. Annotation: Annotation digunakan untuk metadata yang dapat ditambahkan ke kode untuk memberikan informasi tambahan atau mengkonfigurasi perilaku program.

Anotasi dapat digunakan dalam compile-time dan runtime untuk berbagai tujuan seperti dokumentasi, validasi, pengaturan, dan sebagainya.

3. Buat 1 project dengan menerapkan metode pemrograman berorientasi object java, dan berikan keterangan/penjelasan disetiap code program!

Jawab :

```java
package inheritance;

class Kendaraan {
    private int jmlRoda; // atribut privat untuk menyimpan jumlah roda
    private String warna; // atribut privat untuk menyimpan warna

    public void setJmlRoda(int jmlRoda) { // setter untuk mengatur jumlah roda
        this.jmlRoda = jmlRoda;
    }

    public int getJmlRoda() { // getter untuk mendapatkan jumlah roda
        return jmlRoda;
    }

    public void setWarna(String warna) { // setter untuk mengatur warna
        this.warna = warna;
    }

    public String getWarna() { // getter untuk mendapatkan warna
        return warna;
    }
}

// inheritance
class Mobil extends Kendaraan {
    private String bahanBakar; // atribut privat untuk menyimpan bahan bakar
    private int kapasitasMesin; // atribut privat untuk menyimpan kapasitas mesin

    public void setBahanBakar(String bahanBakar) { // setter untuk mengatur bahan bakar
        this.bahanBakar = bahanBakar;
    }

    public String getBahanBakar() { // getter untuk mendapatkan bahan bakar
        return bahanBakar;
    }

    public void setKapasitasMesin(int kapasitasMesin) { // setter untuk mengatur kapasitas mesin
        this.kapasitasMesin = kapasitasMesin;
    }

    public int getKapasitasMesin() { // getter untuk mendapatkan  kapasitas mesin
        return kapasitasMesin;
    }
}

class Truk extends Mobil {
    private int muatanMaks; // atribut privat untuk menyimpan muatan

```

```
49      public void setMuatanMaks(int muatanMaks) { // setter untuk mengatur muatan
50          this.muatanMaks = muatanMaks;
51      }
52
53      public int getMuatanMaks() { // getter untuk mendapatkan muatan
54          return muatanMaks;
55      }
56  }
57
58  class Taksi extends Mobil {
59      private int tarifAwal; // atribut privat untuk menyimpan tarif awal
60      private int tarifPerKm; // atribut privat untuk menyimpan tarif perKm
61
62      public void setTarifAwal(int tarifAwal) { // setter untuk mengatur tarif awal
63          this.tarifAwal = tarifAwal;
64      }
65
66      public int getTarifAwal() { // getter untuk mendapatkan tarif awal
67          return tarifAwal;
68      }
69
70      public void setTarifPerKm(int tarifPerKm) { // setter untuk mengatur tarif perKm
71          this.tarifPerKm = tarifPerKm;
72      }
73
74      public int getTarifPerKm() { // getter untuk mendapatkan tarif perKm
75          return tarifPerKm;
76      }
77  }
78
79  class Sepeda extends Kendaraan {
80      private String jmlSadel; // atribut privat untuk menyimpan jumlah sadel
81      private int jmlGir; // atribut privat untuk menyimpan jumlah gir
82
83      public void setJmlSadel(String jmlSadel) { // setter untuk mengatur jumlah sadel
84          this.jmlSadel = jmlSadel;
85      }
86
87      public String getJmlSadel() { // getter untuk mendapatkan jumlah sadel
88          return jmlSadel;
89      }
90
91      public void setJmlGir(int jmlGir) { // setter untuk mengatur jumlah gir
92          this.jmlGir = jmlGir;
93      }
94
95      public int getJmlGir() { // getter untuk mendapatkan jumlah gir
96          return jmlGir;
```

```
97      }
98  }
99
100 public class Inheritance {
101     public static void main(String[] args) {
102         // exception hendling
103         try {
104             // polimorfis
105             Kendaraan kendaraan = new Taksi(); // membuat objek taksi dari class taksi
106             if (kendaraan instanceof Taksi) {
107                 Taksi taksi = (Taksi) kendaraan;
108                 taksi.setKapasitasMesin(2000);
109
110                 int kapasitasMesinTaksi = taksi.getKapasitasMesin();
111                 System.out.println("Kapasitas Mesin Taksi: " + kapasitasMesinTaksi);
112             }
113         } catch (Exception e) {
114             System.out.println("Terjadi kesalahan: " + e.getMessage());
115         }
116     }
117 }
118
```

LINK GITHUB :
https://github.com/vaclariva/Penerapan-OOP