PRAKTIKUM 2
PEMROGRAMAN BERORIENTASI OBJEK

A. TUJUAN

1. Konsep pemrograman berorientasi objek
2. Menciptakan kelas
3. Membuat objek dari suatu kelas
4. Mengakses variabel dan method dari suatu kelas
5. Kata Kunci this
6. Penentu Akses:public dan private
7. Kata Kunci static

B. DASAR TEORI

- Deklarasi class dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> class <nama_class> {
[deklarasi_atribut]
[deklarasi_konstruktor]
[deklarasi_metode]
}
```

Contoh:

```
public class Siswa
{
  …
}
```

- Deklarasi atribut dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> <tipe> <nama_atribut> ;
```

Contoh:

```
public class Siswa
{
   public int nrp;
   public String nama;
}
```

- Deklarasi metode dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> <return_type> <nama_metode> ([daftar_argumen]) {
    [<statement>]
}
```

Contoh:

```
public class Siswa {
      public int nrp;
      public String nama;
      public void info() {
            System.out.println(“Ini siswa PENS”);
      }
}
```

- Untuk dapat mengakses anggota-anggota dari suatu obyek, maka harus dibuat instance dari class tersebut terlebih dahulu. Berikut ini adalah contoh pengaksesan anggota-anggota dari class Siswa:

```
public class TesSiswa {
public static void main(String args[]) {
      Siswa it=new Siswa();
      it.nrp=5;
      it.nama=”Andi”;
      it.info();
      }
}
```

- Pada pemrograman berorientasi objek, akses terhadap suatu variabel instan diluar kelas biasanya tidak diperkenankan. Sebagai penggantinya, disediakan metode yang diperlukan untuk mengakses variable instant. Berkaitan dengan boleh/tidaknya suatu variable instant diakses dari luar kelas ,Java menyediakan penentu akses. Dua diantara penentu akses yang tersedia adalah private dan public.
  - public berarti bahwa pengaksesan statu variable instan atau metode dapat dilakukan dari luar kelas
  - private berarti bahwa pengaksesan statu variable instan atau metode hanya dapat dilakukan didalam kelas;tidak bisa diakses dari luar kelas

- Kata kunci this sangat berguna untuk menunjukkan suatu member dalam class-nya sendiri. This dapat digunakan baik untuk data member maupun untuk function member, serta dapat juga digunakan untuk konstruktor. Adapun format penulisannya adalah :
  this.data_member → merujuk pada data member
  this.function_member() → merujuk pada function member
  this( ) → merujuk pada konstruktor
  Contoh:

```
public class Siswa
{
      private int nrp;

      public setNrp(int nrp) {
            this.nrp=nrp;
      }
}
```

C. TUGAS PENDAHULUAN

1. Buat project dan class baru, simpan dengan nama tertentu. Semua class pada tugas pendahuluan simpan dalam satu project.
   Amati dan tampilkan hasil dari program dibawah ini!

```
1   class Siswa {
2       int nrp;
3       public void setNRP(int i){
4           nrp=i;
5       }
6   }
7   public class Test {
8       public static void main(String[] args){
9           Siswa anak = new Siswa();
10          anak.setNRP(5);
11          System.out.println(anak.nrp);
12      }
13  }
```

2. Amati dan tampilkan hasil dari program dibawah ini!

```
1   class DataSiswa {
2       int nrp;
3       String nama;
4       public void setNRP(int i){
5           nrp=i;
6       }
7       public void setNama(String j){
8           nama=j;
9       }
10  }
11  public class TestSiswa {
12      public static void main(String[] args){
13          DataSiswa anak = new DataSiswa();
14          anak.setNRP(5);
15          anak.setNama("Budi");
16          System.out.println(anak.nrp);
17          System.out.println(anak.nama);
18      }
19  }
```

Ganti dengan menggunakan NRP sesungguhnya misalkan “1203121061” yang awalnya diberi nilai 5. Apakah terjadi error? Apabila terjadi error, modifikasi program sehingga bisa menggunakan nilai NRP sesungguhnya, dan tampilannya berubah seperti di bawah ini.

```
run:
1203121061
Budi
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Implementasikan UML class diagram di bawah ini dalam bentuk program untuk class Mahasiswa.

Mahasiswa
-nama : String -mata_kuliah : String -nilai : float

Gunakan program di bawah ini untuk pengetesan.

```
6   public class TestMahasiswa {
7       public static void main(String[] args){
8           Mahasiswa anak = new Mahasiswa();
9           anak.nama="Budi";
10          anak.mata_kuliah="Dasar Pemrograman 2";
11          anak.nilai=85;
12          System.out.println(anak.nama+" pada mata kuliah "+anak.mata_kuliah+" mendapatkan nilai "+anak.nilai);
13      }
14  }
```

Hasil yang diharapkan seperti di bawah ini.

```
run:
Budi pada mata kuliah Dasar Pemrograman 2 mendapatkan nilai 85.0
BUILD SUCCESSFUL (total time: 0 seconds)
```