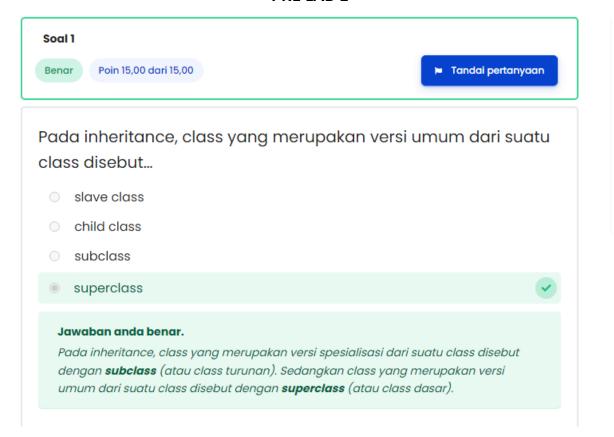
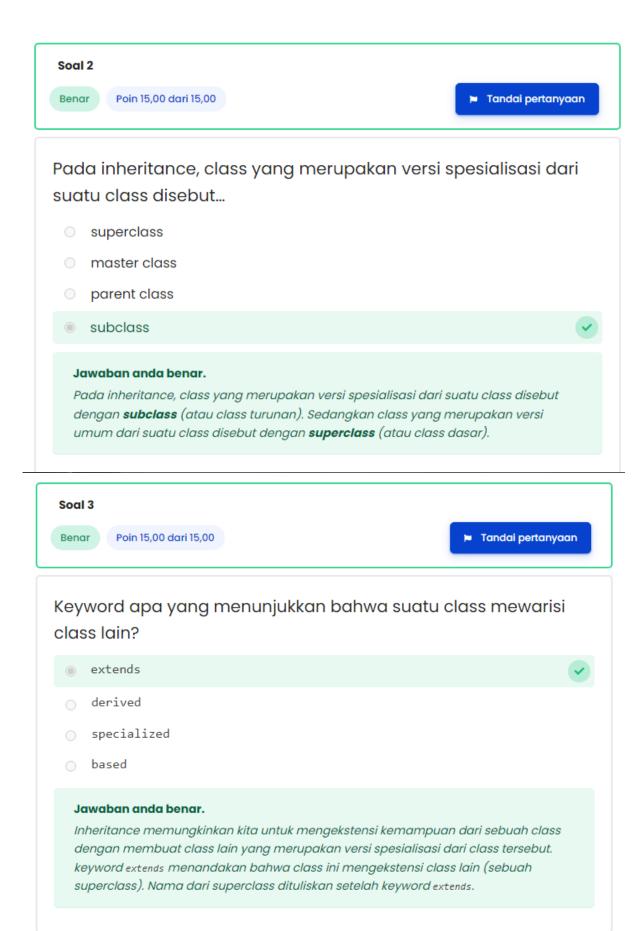
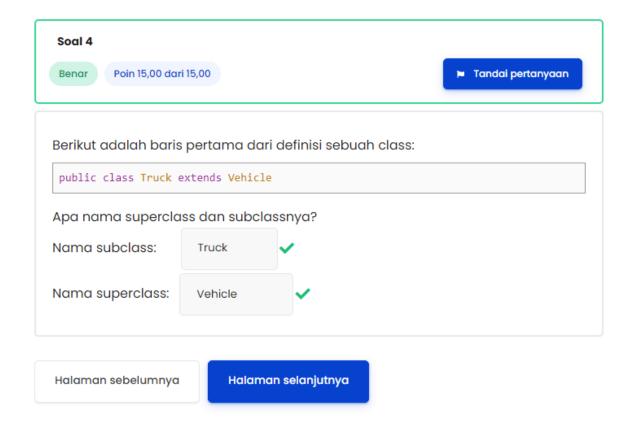
# **PBO TOPIC 9**



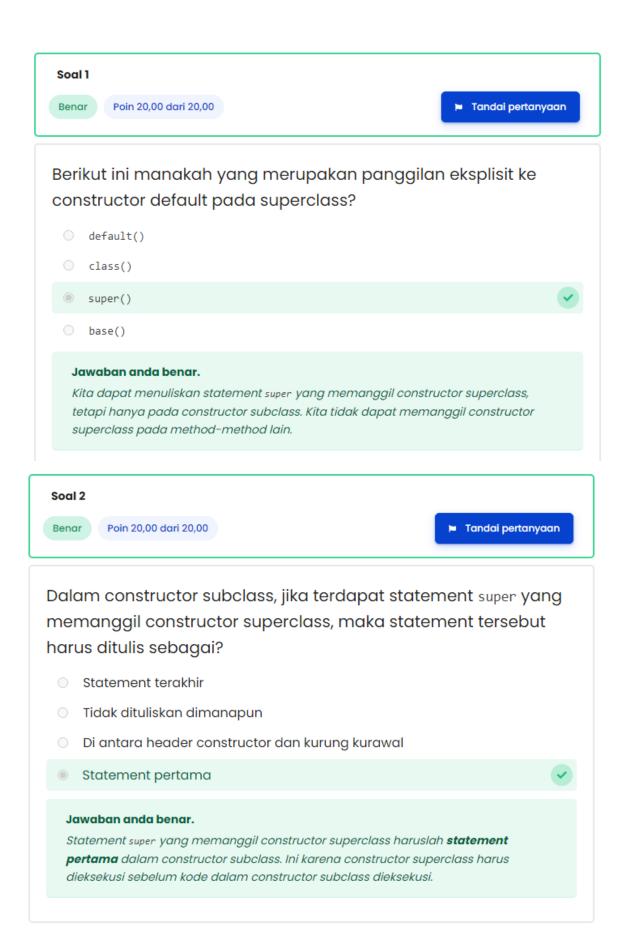




#### No 5

```
public class MobilListrik extends Mobil {
  private int lamaRecharge;
  public MobilListrik(){
    setPabrikan("");
    setModel("");
    setWarna("");
    setKecepatanMax(0);
    this.lamaRecharge = 0;
 }
  public MobilListrik(String pabrikan, String model, String warna, int kecepatanMax, int lamaRecharge){
    setPabrikan(pabrikan);
    setModel(model);
    setWarna(warna);
    setKecepatanMax(kecepatanMax);
    this.lamaRecharge = lamaRecharge;
 }
  public void setLamaRecharge(int lamaRecharge)
    this.lamaRecharge = lamaRecharge;
```

```
public int getLamaRecharge()
    return lamaRecharge;
  }
}
                                                   No 6
public class Lingkaran extends Bangun{
  private double radius;
  public void setRadius(double radius)
    this.radius = radius;
    double luas = Math.PI * radius * radius;
    setLuas(luas);
  }
  public double getRadius()
    return radius;
  }
}
```



```
Car car = new Car(5);
 RaceCar fastCar = new RaceCar(5);
 car.display();
 car.addFuel();
 car.display();
 fastCar.display();
 fastCar.addFuel();
 fastCar.display();
Jawaban:
             5 6 10 11
   Constructor class RaceCar memanggil constructor class Car dengan memberikan
  argument berupa dua kali nilai argument yang diterimanya. Sehingga, statement:
  RaceCar fastCar = new RaceCar(5);
  akan membuat sebuah object RaceCar dengan menginisialisasi field fuel dari
  superclass yang terasosiasikan dengan object tersebut dengan nilai 10.
  Karena class RaceCar mengekstensi class Car, maka semua method public dari class
  Car diwarisi oleh class RaceCar. Ini berarti object dari class RaceCar dapat
  memanggil method-method public dari class Car.
```

#### No 4

```
public class Point3D extends Point2D {
    private int z;

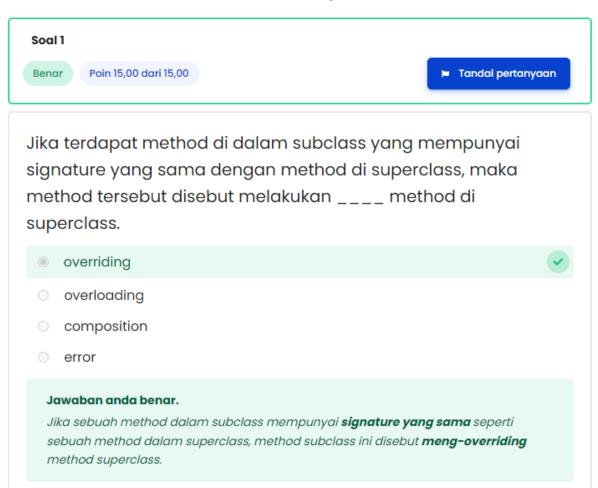
public Point3D()
    {
        super(0, 0);
        this.z = 0;
    }

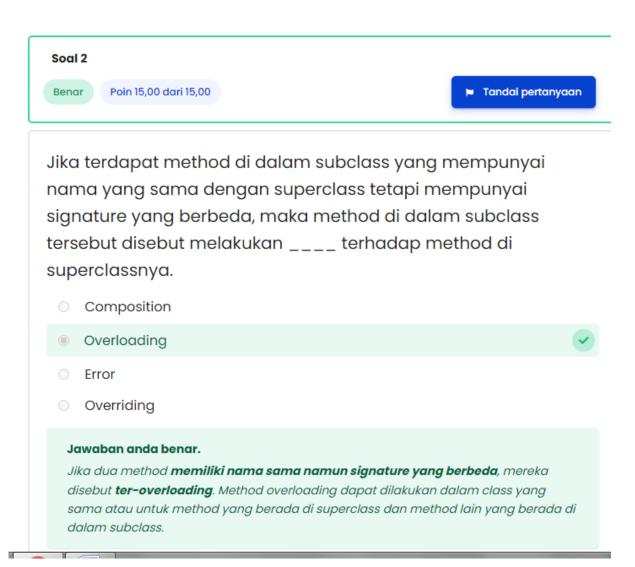
public Point3D(int x, int y, int z) {
        super(x, y);
        this.z = z;
    }

public void setZ(int z)
    {
        this.z = z;
    }

public int getZ()
```

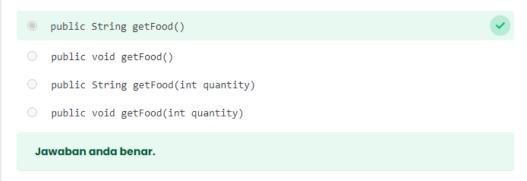
```
{
    return z;
}
}
```





```
public void setId (int theId)
{
    this.id = theId;
}
```

Misalkan kita ingin menambahkan method getFood pada class student yang meng-overriding method getFood pada class Person. Header method getFood manakah dari header-header method di bawah yang akan secara benar meng-overrinding method getFood pada class Person?



```
public class Truk extends Mobil
 {
     public void m1()
         System.out.println("Truk method 1");
 }
Asumsikan terdapat deklarasi-deklarasi variabel berikut:
Mobil myMobil = new Mobil();
 Truk myTruk = new Truk();
Apa output dari statement-statement berikut?
                    Mobil method 1
l. myMobil.m1();
2. myMobil.m2();
                    Mobil method 2
3. myTruk.m1();
                    Truk method 1
4. myTruk.m2();
                    Mobil method 2
```

5. public class Manager extends Pegawai {

4.

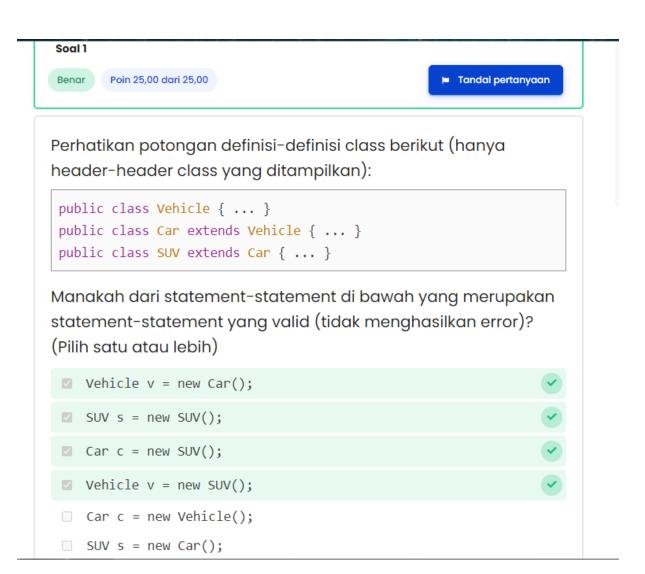
}

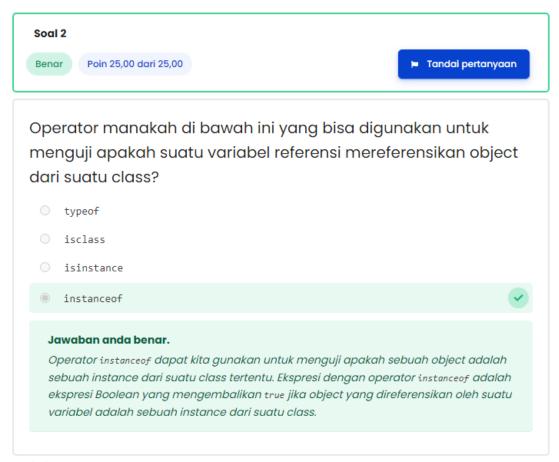
```
private double bonus;

public Manager(String nama, double gajiPokok, double bonus){
    super(nama, gajiPokok);
    this.bonus = bonus;
}

@Override
public double getGaji()
{
    return super.getGaji() + bonus;
}
```

```
    public class BujurSangkar extends PersegiPanjang {
        public BujurSangkar(double sisi) {
            super(sisi, sisi);
            setLuas(getPanjang() * getLebar());
        }
    }
```

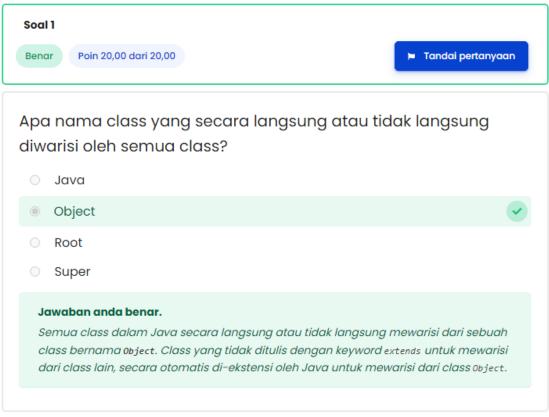




### 3. Prelab 9.5 no 3

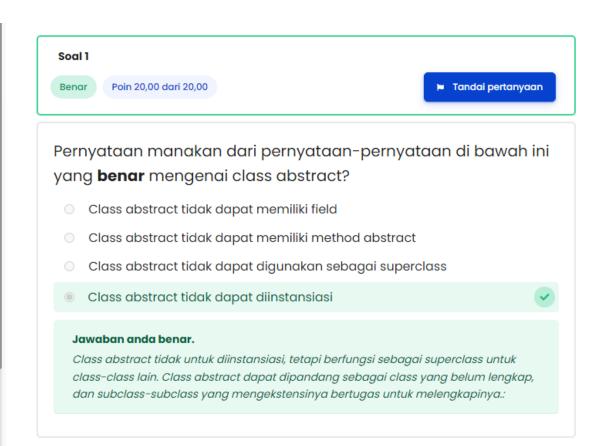
```
System.out.println();
      // Cetak Judul nomor bangun.
       System.out.println("Bangun " + (i + 1) + ": ");
      // Cetak luas bangun
       System.out.println(
"Luas = " + obj[i].getLuas()
);
      System.out.print("Jenis: ");
      // Statement if untuk menguji class dari setiap object
      // dan menampilkan informasi setiap object.
      // Case 1: Jika object adalah dari class PersegiPanjang
      if (
obj[i] instanceof PersegiPanjang
         // Cetak Persegi Panjang
         System.out.println("Persegi Panjang");
         // Casting ke tipe PersegiPanjang
         PersegiPanjang p =
(PersegiPanjang) obj[i];
         // Cetak panjang dan lebar
         System.out.println(
"panjang = " + p.getPanjang() + ", " + "lebar= " + p.getLebar()
);
      // Case 2: Jika object adalah dari class BujurSangkar
      else if (
obj[i] instanceof BujurSangkar
         // Cetak Bujur Sangkar
         System.out.println("Bujur Sangkar");
         // Casting ke tipe BujurSangkar
         BujurSangkar b =
(BujurSangkar) obj[i];
         // Cetak sisi
         System.out.println(
"panjang = " + b.getPanjang() + ", " + "lebar=" + b.getPanjang()
);
      // Case 3: Jika object adalah dari class Lingkaran
      else if (
obj[i] instanceof Lingkaran
```

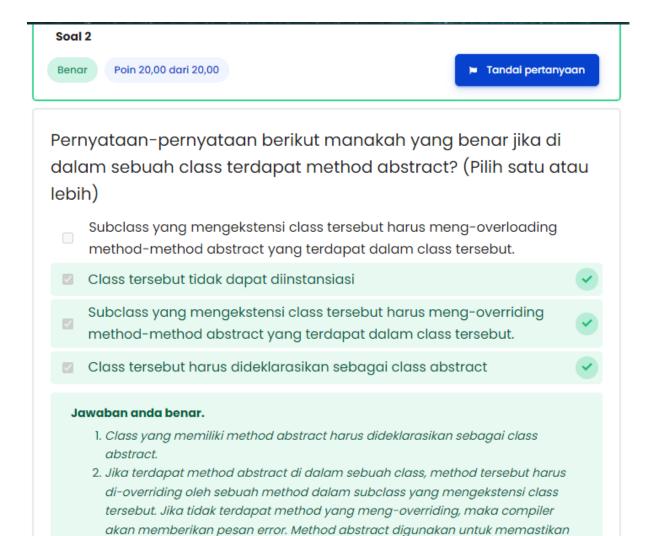
```
)
         // Cetak Lingkaran
         System.out.println("Lingkaran");
         // Casting ke tipe Lingkaran
         Lingkaran I =
(Lingkaran) obj[i];
         // Cetak radius
         System.out.println(
"radius = " + l.getRadius()
);
      }
      // Case 4: Jika object adalah dari class SegitigaSiku
       else if (
obj[i] instanceof SegitigaSiku
       {
         // Cetak Segitiga Siku
         System.out.println("Segitiga Siku");
         // Casting ke tipe SegitigaSiku
         SegitigaSiku s =
(SegitigaSiku) obj[i];
         // Cetak alas dan tinggi
         System.out.println(
"alas = " + s.getAlas() + ", " + "tinggi = " + s.getTinggi()
);
      }
    }
  }
```



```
2
   public class RekeningBank
     private String nomorRekening;
     private double saldo;
      public RekeningBank(String nomorRekening, double saldo)
        this.nomorRekening = nomorRekening;
        this.saldo = saldo;
     }
     public String getNomorRekening()
     {
        return nomorRekening;
     }
     public double getSaldo()
        return saldo;
     }
     // [1] Tambahkan method toString di bawah.
      public String toString(){
        return this.nomorRekening + ", " + this.saldo;
```

```
// [2] Tambahkan method equals di bawah.
public boolean equals(RekeningBank rekening){
   if(rekening.nomorRekening == this.nomorRekening && rekening.saldo == this.saldo){
     return true;
   }
   return false;
}
```





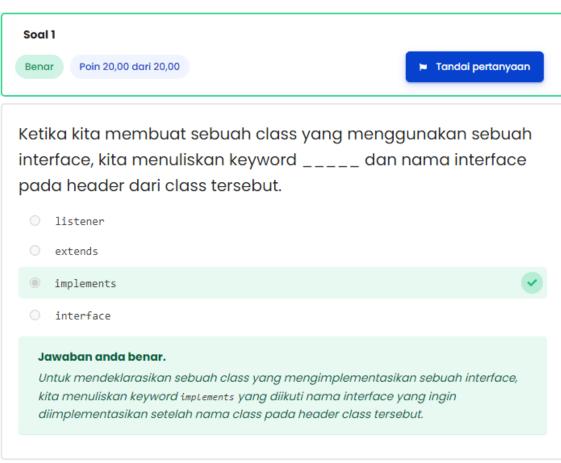
subclass mengimplementasikan (menuliskan kode-kode untuk) method

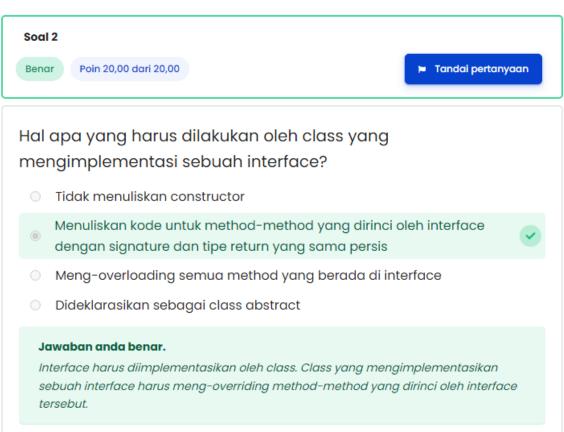
```
public class Kucing extends Hewan{
   public Kucing(){
      super(4);
   }

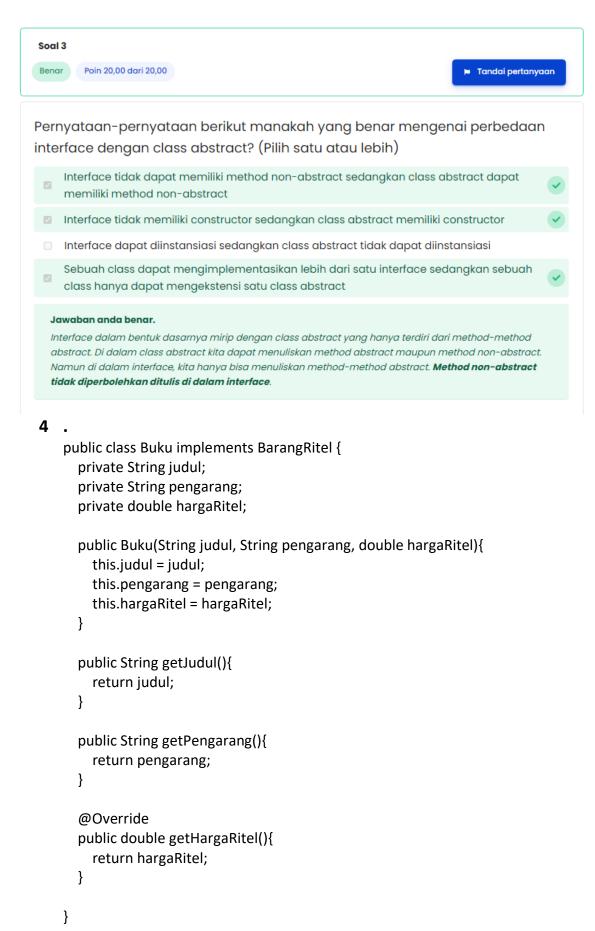
   @Override
   public void bersuara(){
      System.out.println("Meong...");
   }
```

tersebut.

}







```
1.
```

}

```
public class PegawaiHonorer extends Pegawai {
  private double honorPerJam, jamKerja;
  public PegawaiHonorer(String nama, double honorPerJam, double jamKerja){
    super(nama, honorPerJam);
    this.honorPerJam = honorPerJam;
    this.jamKerja = jamKerja;
  }
  @Override
  public double getGaji(){
    return honorPerJam * jamKerja;
  }
}
2.
public class RekeningMinMax extends RekeningBank {
  private double saldoMax, saldoMin;
  public RekeningMinMax(String rekening, double saldoAwal){
    super(rekening, saldoAwal);
    this.saldoMax = saldoAwal;
    this.saldoMin = saldoAwal;
  }
  public double getSaldoMax(){
    return saldoMax;
  }
  public double getSaldoMin(){
    return saldoMin;
  }
  @Override
  public void deposit(double jumlah)
  {
    setSaldo(getSaldo() + jumlah);
    if(getSaldo() > this.saldoMax){
      saldoMax = getSaldo();
```

```
@Override
public void withdraw(double jumlah)
{
    if (getSaldo() >= jumlah)
    {
        setSaldo(getSaldo() - jumlah);

        if(getSaldo() < this.saldoMin){
            saldoMin = getSaldo();
        }
     }
     else
     {
        System.out.println("Dana tidak mencukupi.");
     }
}</pre>
```