1. Dari kode berikut run output dan Tambahkan kode supaya bisa memanggil isiBensin() untuk k1 dan kayuh() untuk k2.

```
class Kendaraan {
        void jalan() { System.out.println("Kendaraan berjalan"); }
}
class Mobil extends Kendaraan {
        @Override
        void jalan() { System.out.println("Mobil berjalan di jalan raya"); }
        void isiBensin() { System.out.println("Isi bensin di SPBU"); }
}
class Sepeda extends Kendaraan {
        @Override
        void jalan() { System.out.println("Sepeda dikayuh di jalan kecil"); }
        void kayuh() { System.out.println("Mengayuh pedal sepeda"); }
}
public class Main {
        public static void main(String[] args) {
    Kendaraan k1 = new Mobil();   // A
    Kendaraan k2 = new Sepeda();  // B
    k1.jalan();   // C
    k2.jalan();   // D
        }
}
```

2. Dari kode berikut Ubah class Hewan menjadi **abstract class.**
Jadikan suara() sebagai abstract method.
Pastikan Kucing dan Anjing tetap bisa berjalan dengan override method masing-masing.

```
class Hewan {

        void suara() {

        System.out.println("Hewan bersuara...");

        }

}

class Kucing extends Hewan {

        void suara() {

        System.out.println("Meong");

        }

}

class Anjing extends Hewan {

        void suara() {
```

```
        System.out.println("Guk guk");

        }

}
```

3. Tambahkan kode berikut dengan abstract method baru isiBahanBakar(). Implementasikan method tersebut di class Mobil.
Buat subclass baru Sepeda yang override jalan(), tapi di isiBahanBakar() tampilkan pesan “Sepeda tidak membutuhkan bahan bakar”.

```
abstract class Kendaraan {
        abstract void jalan();
}
class Mobil extends Kendaraan {
        @Override
        void jalan() {
        System.out.println("Mobil berjalan di jalan raya");
        }
}
```

4. Perhatikan kode berikut

```
abstract class Payment {
        abstract void pay(double amount);
}

class CashPayment extends Payment {
        @Override
        void pay(double amount) {
        System.out.println("Bayar tunai: " + amount);
        }
}
```

Tambahkan subclass QrisPayment yang override pay() dengan output:

```
Bayar via QRIS sejumlah <amount>
```

Di main(), buat array Payment[] daftar = { new CashPayment(), new QrisPayment() };
Gunakan loop untuk memanggil pay() polymorphically.

5. Ubah desain berikut agar menggunakan abstract class Shape dengan abstract method double luas(). Tambahkan subclass Lingkaran (punya atribut jariJari) dan implementasi luasnya. Di main(), buat array Shape[] daftar berisi Persegi dan Lingkaran, lalu tampilkan luas tiap shape dengan loop polymorphism.

```
class Persegi {
        double sisi;
        Persegi(double sisi) { this.sisi = sisi; }
        double luas() { return sisi * sisi; }
```

```
}
```

6. Dari rancangan berikut

```
abstract class Payment {
        protected double amount;
   Payment(double amount) { this.amount = amount; }
        abstract void pay();
}
```

Tambahkan subclass CashPayment, QrisPayment, dan TransferPayment.

CashPayment → “Bayar tunai sejumlah X”

QrisPayment → “Bayar QRIS sejumlah X”

TransferPayment → “Bayar transfer sejumlah X” + method tambahan cekKodeBank().

Simpan semua pembayaran di Payment[] daftar.

Loop untuk memanggil pay().

Gunakan downcasting untuk memanggil cekKodeBank() hanya jika object TransferPayment.