オブジェクト指向プログラミング

学籍番号:22120
組番号:222
名前: 塚田 勇人

2024 年 11 月 25 日

1　ソースコード

リスト 1: Main.java

```
1  public class Main {
2      public static void main(String args[]) {
3          System.out.println("===バス===");
4          Bus bus = new Bus();
5          bus.setGas();
6          bus.setTire(6);
7          bus.setNenpi();
8          bus.drive();
9          bus.drive();
10         bus.drive();
11         bus.drive();
12         System.out.println("===救急車===");
13         Ambulance kyukyu = new Ambulance();
14         kyukyu.setTire(4);
15         kyukyu.setGas();
16         kyukyu.setNenpi();
17         kyukyu.callSiren();
18         System.out.println("===パトカー===");
19         PatrolCar patrol = new PatrolCar();
20         patrol.setTire(4);
21         patrol.setGas();
22         patrol.setNenpi();
23         patrol.callSiren();
24         patrol.drive();
25         patrol.callSiren();
26         patrol.drive();
27         patrol.callSiren();
28         patrol.drive();
29         patrol.callSiren();
30         patrol.drive();
31     }
32 }
```

リスト 2: Ambulance.java

```
public class Ambulance extends Car implements Siren {
  public Ambulance() {
    System.out.println("救急車製造");
  }
  //インターフェースを実装して、救急車のサイレンを鳴らす。
  public void callSiren() {
    System.out.println("ピーポーピーポー");
  }
  //燃料の設定を行う
  public void setGas() {
    this.gas = 120;
    System.out.println("搭載燃料: " + getGas());
  }
  //燃費の設定を行う。
  public void setNenpi() {
    this.nenpi = 5;
    System.out.println("燃費: " + getNenpi());
  }
}
```

リスト 3: Car.java

```
public abstract class Car {
    protected int tire;
    protected int gas;
    protected int nenpi;

    // コンストラクタ
    Car() {
        System.out.println("車製造");
    }

    // コンストラクタのオーバーロード
    Car(int gas, int nenpi, int tire) {
        System.out.println("車作成中");
    }

    // タイヤの設定。異常な場合には補正する
    public void setTire(int tire){
        System.out.println("タイヤ:" + tire);
        if (tire < 3) {
            System.out.println("タイヤの数が少なすぎます。4本に設定します。");
            this.tire = 4;
        } else {
            this.tire = tire;
        }
    }
    //ガスの設定を抽象メソッドとして定義
    public abstract void setGas();
```

```
28      //燃費の設定を抽象メソッドとして定義
29      public abstract void setNenpi();
30      //タイヤ変数を取得
31      public int getTire() {
32          return tire;
33      }
34      //ガス変数を取得
35      public int getGas() {
36          return gas;
37      }
38      //燃費変数を取得
39      public int getNenpi() {
40          return nenpi;
41      }
42      //実行するたびに燃料を減らして車を動かし、残りの燃料を表示させる
43      public void drive() {
44          // setGas(getGas() - nenpi);
45          gas -= nenpi;
46          if(getGas() < 0) {
47              System.out.println("ガス欠です!うごけません! ");
48              return;
49          }
50          System.out.println("ぶいーん 残燃料: " + getGas());
51      }
52
53  }
```

リスト 4: Bus.java

```
1   public class Bus extends Car {
2     int passenger;
3     public Bus() {
4       System.out.println("バス製造");
5     }
6
7     //乗客の設定を行う
8     public void setPassenger() {
9       this.passenger = 0;
10      System.out.println("搭乗人数: " + passenger);
11    }
12    //燃料の設定を行う
13    public void setGas() {
14      this.gas = 20;
15      System.out.println("搭載燃料: " + getGas());
16    }
17    //燃費の設定を行う。
18    public void setNenpi() {
19      this.nenpi = 10;
20      System.out.println("燃費: " + getNenpi());
21    }
```

```
22
23 }
```

リスト 5: PatrolCar.java

```
1  public class PatrolCar extends Car implements Siren {
2    public PatrolCar() {
3      System.out.println("パトカー製造");
4    }
5    //インターフェースを実装して、パトカーのサイレンを鳴らす。
6    public void callSiren() {
7      System.out.println("うおーーーーん");
8    }
9    //燃料の設定を行う
10   public void setGas() {
11     this.gas = 12;
12     System.out.println("搭載燃料: " + getGas());
13   }
14   //燃費の設定を行う。
15   public void setNenpi() {
16     this.nenpi = 5;
17     System.out.println("燃費: " + getNenpi());
18   }
19 }
```

リスト 6: Siren.java

```
1  public interface Siren {
2    public void callSiren() ;
3  }
```

2 実行結果

リスト 7: 実行結果

```
1   ===バス===
2   車製造
3   バス製造
4   搭載燃料: 20
5   タイヤ:6
6   燃費: 10
7   ぶいーん 残燃料: 10
8   ぶいーん 残燃料: 0
9   ガス欠です!うごけません!
10  ガス欠です!うごけません!
11  ===救急車===
12  車製造
13  救急車製造
```

```
タイヤ:4
搭載燃料: 120
燃費: 5
ピーポーピーポー
===パトカー===
車製造
パトカー製造
タイヤ:4
搭載燃料: 12
燃費: 5
うおーーーーん
ぶいーん 残燃料: 7
うおーーーーん
ぶいーん 残燃料: 2
うおーーーーん
ガス欠です!うごけません!
うおーーーーん
ガス欠です!うごけません!
```

3 考察

クラスの継承を用いることで、共通の機能を持つクラスを作成することができる。例えば、車のクラスを作成し、そのクラスを継承してバス、救急車、パトカーのクラスを作成することで、車の共通の機能を持つクラスを作成することができる。
また、インターフェースを用いることで、クラスに共通の機能を持たせることができる。例えば、サイレンのインターフェースを作成し、そのインターフェースを実装することで、サイレンの機能を持つクラスを作成することができる。