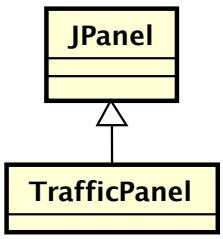
オブジェクト指向プログラミング 令和６年度 後期中間試験

(2024.12.16 重村 哲至) IE5 _____番 氏名_____ 模範解答

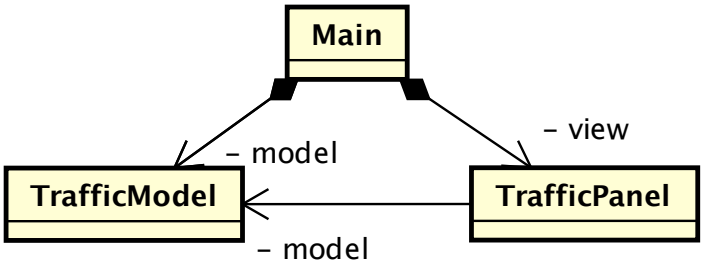
以下では，付録に掲載した交通アプリについて答えなさい．

1 クラス図に関する問題

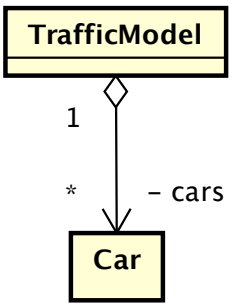
1. 次のクラス図に関連を描き加えなさい．(5 点)



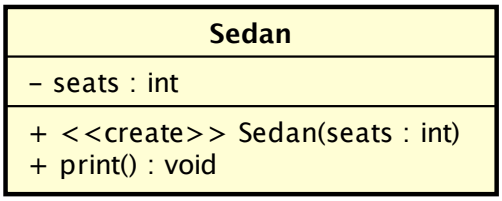
2. 次のクラス図に関連を描き加えなさい．関連には，誘導可能性，集約，ロール名を書くこと．(5 点)



3. 次のクラス図に関連を描き加えなさい．関連には，誘導可能性，集約，ロール名，多重度を書くこと．(5 点)



4. Sedan クラスの，属性（プロパティ），操作（メソッド），コンストラクタを記入したクラス図を描きなさい．コンストラクタはステレオタイプ<<create>>を付けて表現すること．(5 点)



オブジェクト指向プログラミング 令和６年度 後期中間試験

(2024.12.16 重村 哲至) IE5 ＿＿＿番 氏名＿＿＿＿＿＿ 模範解答

2 型に関する問題

付録プログラムのクラスが利用可能なとき，次のプログラムでコンパイル時にエラーになる行を「×」，エラーにならない行を「○」で答えなさい．
(3 点 ×10 問=30 点)

```
Car c1 = new Car("car.png", 10);       // (1)
Car c2 = new Sedan("sedan.png", 50);   // (2)
Car c3 = new Sedan(5);                 // (3)
Car c4 = new Truck(8);                 // (4)
Sedan s1 = new Car("car.png, 50);      // (5)
Sedan s2 = new Sedan("sedan.png",50); // (6)
Sedan s3 = new Sedan(5);               // (7)
Sedan s4 = new Truck(8);               // (8)
TrafficModel t = new TrafficModel();
t.add(new Car("car.png", 10));         // (9)
t.add(new Sedan(5));                   // (10)
```

(1)	○	(2)	×	(3)	○	(4)	○
(5)	×	(6)	×	(7)	○	(8)	×
(9)	○	(10)	○				

3 実行結果

Main クラスのコンストラクタの最後の行「`model.print();`」を実行した結果，表示されるものを以下に書きなさい．(10 点)

セダン(定員4名,時速50km)
トラック(8トン積み,時速30km)
スポーツカー(時速60km)

4 プログラムの穴埋め

1. TrafficModel クラスの`###(1)###`は，自動車の `mileage` を `speed` だけ増やすプログラムです．適切な記述を答えなさい．なお，行が長くなるので適当に改行して書くこと．(5 点)

```
car.setMileage(
  car.getMileage()
  + car.getSpeed()
)
```

2. Car クラスの`###(2)###`は，自動車の `speed` を設定する部分です．適切な記述を答えなさい．(5 点)

```
this.speed = speed
```

3. Sedan, Truck, Coupe クラスの`###(3)###`には同じ記述をします．適切な記述を答えなさい．(5 点)

```
@Override
```

4. Main クラスの`###(4)###`に適切な記述を，同じ行のコメントを参考に答えなさい．(5 点)

```
Sedan(4)
```

5. Main クラスの`###(5)###`に適切な記述を，同じ行のコメントを参考に答えなさい．(5 点)

```
Truck(8)
```

6. Main クラスの`###(6)###`に適切な記述を，同じ行のコメントを参考に答えなさい．(5 点)

```
Coupe()
```

7. Main クラスの`###(7)###`に適切な記述を，同じ行のコメントを参考に答えなさい．(5 点)

```
view.setModel(model)
```

8. Main クラスの`###(8)###`に適切な記述を，同じ行のコメントを参考に答えなさい．(5 点)

```
view.repaint()
```

付録 交通アプリ

画面の様子

交通アプリの動作画面です．３種類の自動車が左から右に走ります．右端に到達すると少し下の左端から右に走ります．自動車は sedan.png(乗用車)，truck.png(トラック)，coupe.png(スポーツカー) の３種類の画像で表示します．

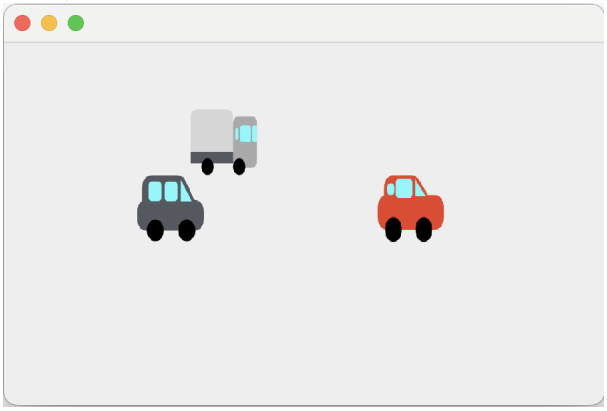


図 1: 交通アプリの表示例

ソースプログラム

リスト 1: TrafficPanel.java

```
... import 省略 ...
public class TrafficPanel extends JPanel {
  private static final int w = 50;                  // 画像の描画サイズ w
  private static final int h = 50;                  // 画像の描画サイズ h
  private TrafficModel model;                       // 描画すべきモデル
  public void setModel(TrafficModel model) {
    this.model = model;                             // モデルのsetter
  }
  @Override
  protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    if (model!=null) {
      int width = getWidth();                       // 表示領域の横幅
      Vector<Car> cars = model.getCars();
      for (int i=0; i<cars.size(); i++) {           // 全ての配列要素について
        Car car = cars.elementAt(i);                //   自動車を取り出し
        int mileage = car.getMileage();             //   走行距離を調べる
        g.drawImage(car.getShape(),                 //   自動車の画像を
            mileage % (width - w),                  //     描画位置 x 座標
            mileage / (width - w) * h,              //     描画位置 y 座標
            w, h, null);                            //     w × hに収まる大きさで
      }                                             //     描画する
    }
  }
}
```

リスト 2: TrafficModel.java

```
... import 省略 ...
public class TrafficModel {
  private Vector<Car> cars = new Vector<Car>(); // 自動車の可変長配列
  public void tick() {                          // 時刻を進める
    for (int i=0; i<cars.size(); i++) {         //  全ての配列要素について
      Car car = cars.elementAt(i);              //   自動車を取り出し
      ###(1)###;                                //   自動車のmileageを進める
    }
  }
  public void add(Car car) {                    // 自動車を追加する
    cars.add(car);
  }
  public Vector<Car> getCars() {                // 自動車の配列を返す
    return cars;
  }
  public void print() {                         // 全ての自動車の
    for (int i=0; i<cars.size(); i++) {         // 特徴を表示する
      cars.elementAt(i).print();
    }
  }
}
```

リスト 3: Car.java

```
... import 省略 ...
public class Car {                              // 一般的な自動車
  private Image shape;                          // 自動車の画像
  private int speed;                            // 自動車のスピード
  private int mileage = 0;                      // 自動車の走行距離
  public Car(String fname, int speed) {         // コンストラクタ
    try {
      shape = ImageIO.read(new File(fname));    // 画像をファイルから読み込む
    } catch (IOException e) {
      shape = null;
      e.printStackTrace();
    }
    ###(2)###;                                  // 自動車のスピードを設定する
  }
  public Image getShape() { return shape; }     // shape の getter
  public int getSpeed() { return speed; }       // speed の getter
  public void setMileage(int mileage) {         // mileage の setter
    this.mileage = mileage;
  }
  public int getMileage() { return mileage; } // mileage の getter
  public void print() {                         // この自動車の特徴を表示
    System.out.println("種類不明の自動車");    // 種類不明？？
  }                                             // そんな自動車はあり得ない！！
}
```

リスト 4: Sedan.java

```
public class Sedan extends Car {                    // 乗用車クラス
  private int seats;                                // 乗車定員
  public Sedan(int seats) {                         // seats人乗りの乗用車を作る
    super("sedan.png", 50);                         // 時速50kmで走る
    this.seats = seats;
  }
  ###(3)###
  public void print() {                             // セダンの特徴を表示する
    System.out.printf(
        "セダン(定員%d名,時速%dkm)\n",
        seats, getSpeed());
  }
}
```

リスト 5: Truck.java

```
public class Truck extends Car {                    // トラッククラス
  private int ton;                                  // 積載量（トン）
  public Truck(int ton) {
    super("truck.png", 30);                         // 時速30kmで走る
    this.ton = ton;
  }
  ###(3)###
  public void print() {                             // トラックの特徴を表示する
    System.out.printf(
        "トラック(%dトン積み,時速%dkm)\n",
        ton, getSpeed());
  }
}
```

リスト 6: Coupe.java

```
public class Coupe extends Car {  // スポーツカークラス
  public Coupe() {
    super("coupe.png", 60);       // 時速60kmで走る
  }
  ###(3)###
  public void print() {           // スポーツカーの特徴を表示する
    System.out.printf("スポーツカー(時速%dkm)\n",
        getSpeed());
  }
}
```

リスト 7: Main.java

```
... import 省略 ...
public class Main {
  private JFrame frame;
  private TrafficModel model = new TrafficModel();   // モデル
  private TrafficPanel view;                         // ビュー
  public static void main(String[] args) {           // WindowBuilderが生成した
    EventQueue.invokeLater(new Runnable() {          // いつもの main メッソド
      public void run() {
        try {
          Main window = new Main();
          window.frame.setVisible(true);
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    });
  }
  public Main() {                                    // コンストラクタ
    initialize();                                    // 画面を表示する
    Sedan sedan = new ###(4)###;                     // 4人乗りの乗用車
    Truck truck = new ###(5)###;                     // 8トン積みのトラック
    Coupe coupe = new ###(6)###;                     // スポーツカー
    model.add(sedan);
    model.add(truck);
    model.add(coupe);
    ###(7)###;                                       // ビューにモデルを登録する
    new Timer(500, new ActionListener() {            // 500ms 毎に
      public void actionPerformed(ActionEvent evt) {
        model.tick();                                // モデルの時刻を進め
        ###(8)###;                                   // 表示を更新する
      }
    }).start();                                      // タイマースタート
    model.print();                                   // 全自動車の特徴を表示
  }
  private void initialize() {                        // WindowBuilderが作成した
    frame = new JFrame();                            // いつもの initialize
    frame.setBounds(100, 100, 450, 300);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    view = new TrafficPanel();
    frame.getContentPane().add(view, BorderLayout.CENTER);
  }
}
```