オブジェクト指向プログラミング H27 年度 後期中間試験　(2015.11.26 重村 哲至)

IE5　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

1. 適切な用語を答えなさい。(3 点× 5 問=15 点)

(1) クラスを継承して作る新しいクラスのこと（XX クラス）

サブクラス

(2) 継承のもとになったクラスのこと（(1) の逆だね）

スーパークラス

(3) オブジェクト内部の構造や仕様を外部から隠蔽し、オブジェクトの操作は公開メソッドだけを通して行うようにオブジェクトを作ること

カプセル化

(4) (1) と (2) のクラスでは、どちらがどちらの代用として使用できるか？

サブクラスがスーパークラスの代用として使用できる

(5) オブジェクト指向プログラミングでは、オブジェクトは相互に何を送り合うと考えるか？

メッセージ

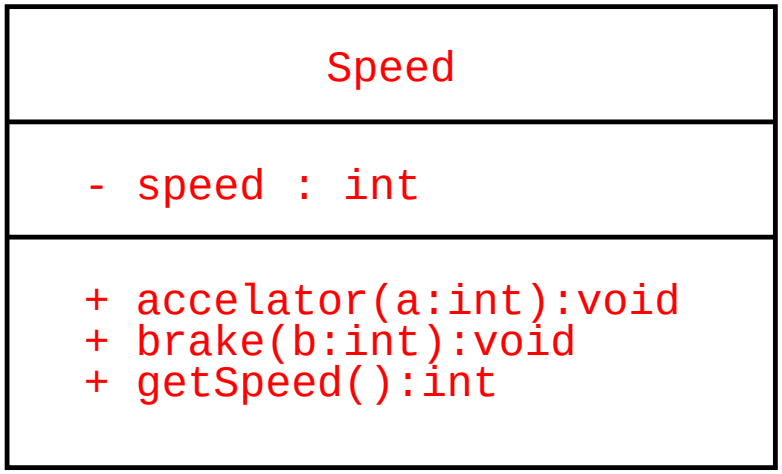
2. 次の Java クラスは自動車のスピードを管理します。

```
class Speed {
  private int speed; // スピードを km/h で記憶
  public void accelator(int a) {...}
  public void brake(int b) {...}
  public int getSpeed() {...}
}
```

- スピードは 0〜100[km/h] の範囲とします。
- `void accelatator(int a)` はアクセルを踏む操作を表しスピードを `a`[km/h] 増加させます。`a` の値が負の場合は何もしないことにします。
- `void brake(int b)` はブレーキを踏む操作を表しスピードを `b`[km/h] 減少させます。`b` の値が負の場合は何もしないことにします。
- `int getSpeed()` は現在のスピードを取り出します。

以上の条件を守るように気をつけて、クラス図と 3 つのメソッドを完成しなさい。

(1) Speed クラスのクラス図を完成しなさい。(10 点)



(2) accelator() を書きなさい。(5 点)

```
public void accelator(int a) {
  if (a>0) {
    speed+=a;
    if (a>100) a = 100;
  }
}
```

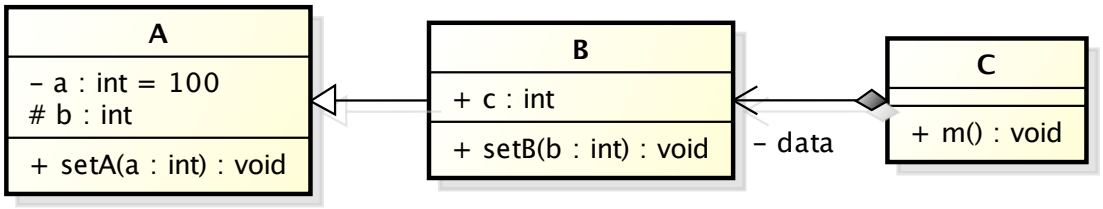
(3) brake() を書きなさい。(5 点)

```
public void brake(int b) {
  if (b>0) {
    speed-=b;
    if (b<0) b = 0;
  }
}
```

(4) getSpeed() を書きなさい。(5 点)

```
public int getSpeed() {
  return speed;
}
```

3. クラス図から Java プログラムを完成しなさい。(10 点× 3 問=30 点)



`setA()`、`setB()` は、単純に対応する変数の値を変更する。
`m()` は、`setA(10)` メッセージをオブジェクトへ送信する。
合成集約の Java プログラムは、別紙「色を作るアプリ」の `ColorApp` クラス内部で、変数 `model`、`view` を初期化している部分を参考にしなさい。その他でも、別紙のクラス図とプログラムを参考にしなさい。

```
public class A {

  private int a = 100;
  protected int b;
  public void setA(int a) { this.a = a; }

}

public class B

              extends A {
  public int c;
  public void setB(int b) { this.b = b; }

}

public class C {

  B data = new C();
  public void m() {
    data.setA(10);
  }

}
```

4. 別紙「色を作るアプリ」は、赤成分 (0〜255)、緑成分 (0〜255)、青成分 (0〜255) の 3 つの値を入力して色を作り中央のパネルの背景色として表示すると同時に、下部のテキストフィールドに 3 つの値をカンマ区切りで例えば”255,255,255” のように表示するアプリです。

(1) /*** (A) ***/ 部分に適切な 1 行のプログラムを書きなさい。(5 点)

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

(2) /*** (B) ***/ 部分に適切な複数行のプログラムを書きなさい。(10 点)

```
try {
     model.setG(Integer.parseInt(valG.getText()));
} catch (NumberFormatException e1) {
     model.setG(0);
}
valG.setText(model.getG()+"");
view.repaint();
```

(3) /*** (C) ***/ 部分に適切な 1 行のプログラムを書きなさい。(5 点)

```
model = m;
```

(4) /*** (D) ***/ 部分に適切な 1 行のプログラムを書きなさい。(5 点)

```
repaint();
```

(5) /*** (E) ***/ 部分に適切な 1 行のプログラムを書きなさい。(5 点)

```
return model;
```

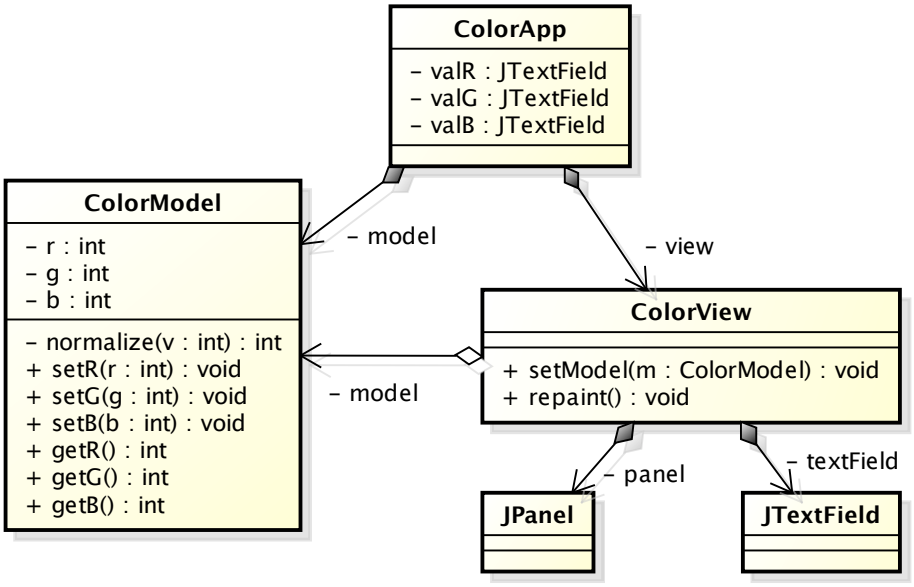


図 1 色を作るアプリのクラス図

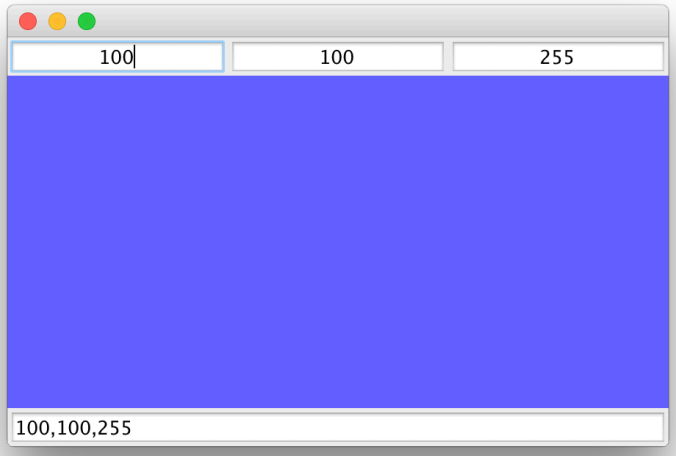


図 2 色を作るアプリの実行例

リスト 1: ColorApp.java

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class ColorApp {
    private JFrame frame;
    private JTextField valR;
    private JTextField valG;
    private JTextField valB;
    private ColorView view;
    private ColorModel model = new ColorModel();   // モデル

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    ColorApp window = new ColorApp();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public ColorApp() {
        initialize();
        /*** (A) ***/      // ColorModel のインスタンスを ColorView に登録する
    }

    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel btnPanel = new JPanel();
        frame.getContentPane().add(btnPanel, BorderLayout.NORTH);
        btnPanel.setLayout(new GridLayout(0, 3, 0, 0));

        // 赤成分を入力する左側のテキストフィールド
        valR = new JTextField();
        valR.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) { // 赤成分値が入力された
                try {
                    model.setR(Integer.parseInt(valR.getText()));
                } catch (NumberFormatException e1) {
                    model.setR(0);                        // 入力エラーなら 0
                }
                valR.setText(model.getR()+"");            // エラー時は 0 を表示
                view.repaint();                           // モデル変化時に再表示
            }
        });
```

```
        valR.setHorizontalAlignment(SwingConstants.CENTER);
        valR.setText("255");
        btnPanel.add(valR);
        valR.setColumns(10);

        // 緑成分を入力する中央のテキストフィールド
        valG = new JTextField();
        valG.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) { // 緑成分値が入力された
                /*** (B) ***/                            // 緑成分に付いて赤成分と同様な処理をする
            }
        });
        valG.setHorizontalAlignment(SwingConstants.CENTER);
        valG.setText("255");
        btnPanel.add(valG);
        valG.setColumns(10);

        // 青成分を入力する右のテキストフィールド
        valB = new JTextField();
        valB.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) { // 青成分の値が入力された
                // ... 省略 ...
            }
        });
        valB.setHorizontalAlignment(SwingConstants.CENTER);
        valB.setText("255");
        btnPanel.add(valB);
        valB.setColumns(10);

        // 色を、表示色と''255,255,255''のようなテキストで表示するパネル
        view = new ColorView();
        frame.getContentPane().add(view, BorderLayout.CENTER);
    }
}
```

リスト 2: ColorModel.java

```
public class ColorModel {
    private int r = 255;                                // 赤成分
    private int g = 255;                                // 緑成分
    private int b = 255;                                // 青成分

    private int normalize(int v) {                      // 入力値を
        if (v<0) v=0;                                   //   0 から 255 の
        else if (v>255) v=255;                          //     範囲に制限
        return v;
    }

    public void setR(int r) { this.r=normalize(r); }    // setter は
    public void setG(int g) { this.g=normalize(g); }    //   値を正常値に制限
    public void setB(int b) { this.b=normalize(b); }    //     してセットする
    public int getR() { return r; }                     // getter は
    public int getG() { return g; }                     //   値をそのまま
    public int getB() { return b; }                     //     返すだけ
}
```

リスト 3: ColorView.java

```
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JTextField;

@SuppressWarnings("serial")
public class ColorView extends JPanel {
    private ColorModel model;
    private JPanel panel;
    private JTextField textField;

    public ColorView() {
        setLayout(new BorderLayout(0, 0));
        panel = new JPanel();
        add(panel, BorderLayout.CENTER);       // 中央に色表示パネルを配置
        textField = new JTextField();
        add(textField, BorderLayout.SOUTH);    // 南に''255,255,255''のように
        textField.setColumns(10);              //   色を表示するテキスト
    }                                          //     フィールドを配置

    public void setModel(ColorModel m) {       // モデルの setter
        /*** (C) ***/                          // 新しいモデルをセットする
        /*** (D) ***/                          // モデルが変更されたら再表示
    }

    public ColorModel getModel() {             // モデルの getter
        /*** (E) ***/
    }

    @Override
    public void repaint() {                    // 再表示
        if (model!=null) {                     // モデルがセットされているか
            int r = model.getR();              //   されているならモデルの
            int g = model.getG();              //     3色成分を取り出し
            int b = model.getB();
            textField.setText(r+","+g+","+b);  // 数値で表示
            panel.setBackground(new Color(r,g,b)); // パネルの色で表示
        }
        super.repaint();                       // repaint イベントを発行
    }
}
```