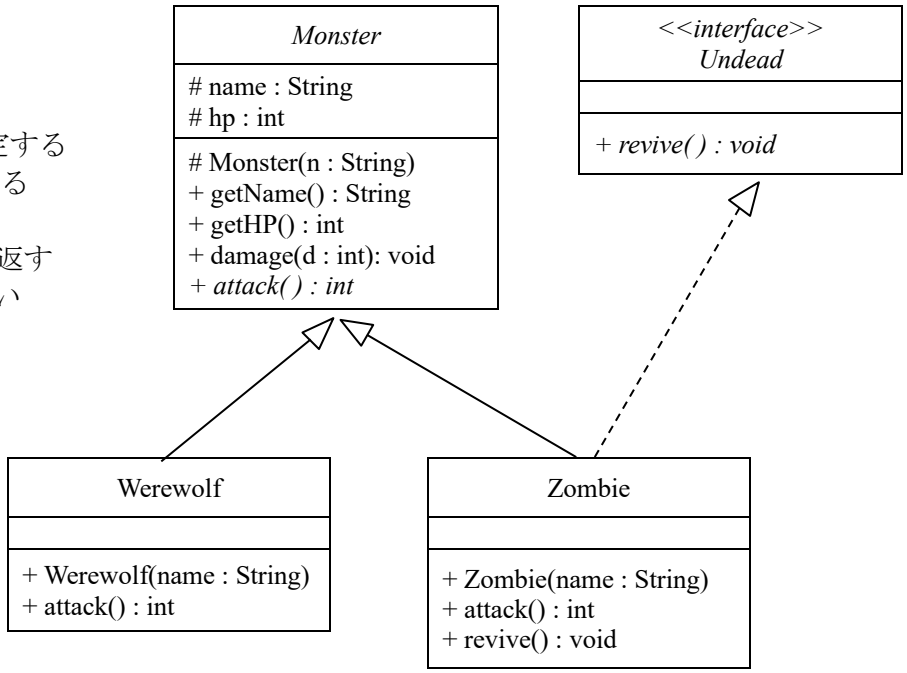
プログラミングⅡ　2022　第 12 回　演習課題　「インタフェース」

1. 下記の UML のクラス図で示した Monster（怪物），Werewolf（人狼），Zombie（ゾンビ）のクラスと Undead（不死）のインタフェースを Java で作成せよ。さらに，main メソッドとそれを含むクラスも作成して，出力結果を示せ。以下にフィールドとメソッドの仕様を簡単に説明する。

説明

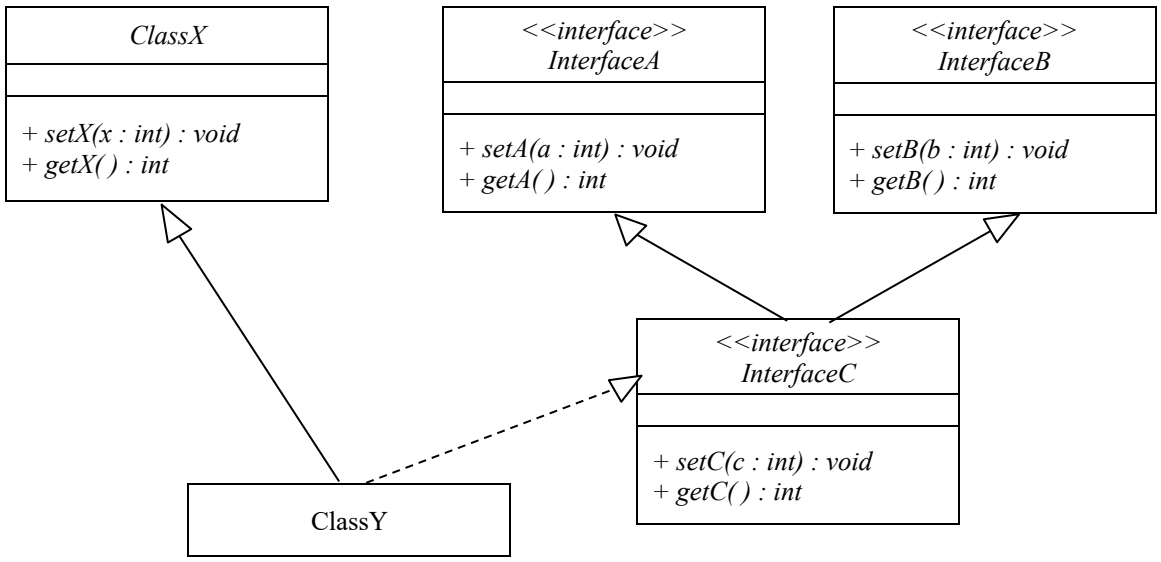
name : モンスターの名前
hp : 体力
Monster() : 名前を設定する
Werewolf() : 体力を 150 に設定する
Zombie() : 体力を 50 に設定する
damage() : 体力を d 減らす
attack() : 攻撃値として整数を返す
　　値は適当に決めてよい
revive() : 体力を元に戻す

※ 斜体に注意すること




```
/* MonsterBattle.java */
import java.util.Random;

public class MonsterBattle {
    public static void main(String[] args) {
        Random rand= new Random();

        Monster [] monsters = { new Werewolf("人狼"), new Zombie("ゾンビ") };

        for (int i = 0; i < 30; i++) {
            System.out.println(monsters[0].getName() + " " + monsters[0].getHP()
                + "  " + monsters[1].getName() + " " + monsters[1].getHP());

            Monster offense = monsters[i % 2];
            Monster defense = monsters[(i + 1) % 2];

            int points = rand.nextInt(offense.attack());
            defense.damage(points);
            System.out.println(defense.getName() + "に" + points + "のダメージ。");
            if (defense.getHP() <= 0) {
                if (defense instanceof Zombie) {
                    ((Zombie)defense).revive();
                    System.out.println(defense.getName() + "は生き返った！");
                } else {
                    System.out.println(defense.getName() + "は死んだ。");
                    return;
                }
            }
        }
        System.out.println("引き分け！");
    }
}
```

2. 下記の UML のクラス図で示した抽象クラス ClassX とインタフェース InterfaceA，InterfaceB，InterfaceC を Java で定義した上で, 図中では省略したフィールドとメソッドを補ってクラス ClassY を定義せよ。さらに, main メソッドも作成して ClassY の動作を確認し，ClassY の完全なクラス図も示せ。




```
/* Main.java */

public class Main {

    public static void main(String[] args) {

        ClassY y = new ClassY();

        y.setX(1);
        System.out.println(y.getX());

        y.setA(2);
        System.out.println(y.getA());

        y.setB(3);
        System.out.println(y.getB());

        y.setC(4);
        System.out.println(y.getC());
    }

}
```

3. 下記は Swing という Java の標準 GUI ライブラリの使用例である。Swing では，ユーザが画面上のボタンを押すと， ActionListener インタフェースの actionPerformed メソッドが実行されるしくみがある。そのためには，事前にボタンの addActionLister メソッドで ActionListener インタフェースを実装したインスタンスを登録しておく必要がある。空欄を適切に埋めて実行させ，スクリーンショットを提出せよ。
※ コンパイルに失敗する際は，別の JDK のインストールが必要になる可能性があるので相談すること。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyApplication [ 空欄(1) ] ActionListener {
   // ActionListener インタフェースを実装すると，ボタンが押されたときの通知先になることができる

   private int x = 0;                                  // 表示する数値

   private JFrame fr;                                  // ウィンドウを表すクラスの変数
   private JLabel lb;                                  // 文字列ラベルを表すクラスの変数
   private JButton bt1, bt2;                           // ボタンを表すクラスの変数
   private Font fn = new Font("Sans-serif", Font.PLAIN, 32);  // フォントのインスタンスの生成

   public [ 空欄(2) ] () {                             // コンストラクタ（アプリの画面を構成する）
      this.fr = new [ 空欄(3) ] ("サンプル");          // ウィンドウの生成（ヒント：fr のクラスは？）
      this.fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  // fr を閉じたらプログラム終了
      this.fr.setSize(400, 300);
      this.fr.setLayout(null);                         //  GUI 部品の自動レイアウトを OFF（座標モード）

      this.lb = new [ 空欄(4) ] ("0");                 // ラベルの生成（ヒント：lb のクラスは？）
      this.lb.setFont(fn);                             // ラベル lb のフォントの設定
      this.lb.setBounds(200, 100, 50, 50);             // ラベル lb の位置（座標）とサイズの設定
      this.fr.add(this.lb);                            // ラベル lb をフレーム fr の上に載せる

      this.bt1 = new [ 空欄(5) ] ("増やす");           // ボタンの生成（ヒント：bt1 のクラスは？）
      this.bt1.setBounds(50, 200, 100, 50);
      this.bt1.addActionListener(this);                // ボタンが押されたときの通知先を自分にする
      this.fr.add(this.bt1);

      this.bt2 = new [ 空欄(6) ] ("減らす");           // ボタンをもう 1 つ生成
      this.bt2.setBounds(250, 200, 100, 50);
      this.bt2.addActionListener(this);
      this.fr.add(this.bt2);

      this.fr.setVisible(true);                        // 構成したウィンドウを画面に表示する
   }

   @Override
   public void actionPerformed(ActionEvent e) {  // ボタンが押されたときの処理を実装
      JButton b = ( [ 空欄(7) ] ) e.getSource();  // 押されたボタン（イベントの発生源）を取得
      if ( [ 空欄(8) ] == this.bt1) {             // 押されたのが bt1 なら
         this.x++;
      } else if ( [ 空欄(9) ] == this.bt2) {      // 押されたのが bt2 なら
         this.x--;
      }
      this.lb.setText(String.valueOf(x));          // ラベル lb の表示内容を新しい x の値に変更
   }

   public static void main(String[] args) {        // プログラムの main メソッド（ここから開始）
      new MyApplication();                          // アプリ本体のインスタンスを生成して実行する
   }
}
```