Структури от Данни и Обектно-Ориентирано Програмиране
спец. Компютърни Науки
Упражнение №3
11.03.2010г.

ТЕМА: Абстрактни класове. Интерфейси. Стек

Задача 1: Следната съвкупност от класове описва функционирането на машина с краен брой състояния. Поведението на конкретна машина се задава от нейната програма, която е последователност от състоянията, през които машината преминава.

```
interface State {
        void run();
}

abstract class StateMachine {

        protected State currentState;

        abstract protected boolean changeState();

        protected final void runAll() {
                while(changeState())
                        currentState.run();
        }
}

// A different subclass for each state:
class Wash implements State {
        public void run() {
                System.out.println("Washing");
        }
}

class Spin implements State {
        public void run() {
                System.out.println("Spinning");
        }
}

class Rinse implements State {
        public void run() {
                System.out.println("Rinsing");
        }
}
```

```
class Washer extends StateMachine {

        private int i = 0;
        // The state table:
        private State states[] = {
                        new Wash(), new Spin(),
                        new Rinse(), new Spin(),
        };

        public Washer() { runAll(); }

        public boolean changeState() {
                if(i < states.length) {
                        // Change the state
                        currentState = states[i++];
                        return true;
                } else
        return false;
        }
}

public class StateMachineDemo {

        Washer w = new Washer();

        public void test() {
                System.out.println("Constructor has done all the work!");
        }

        public static void main(String args[]) {
                new StateMachineDemo().test();
        }
}
```

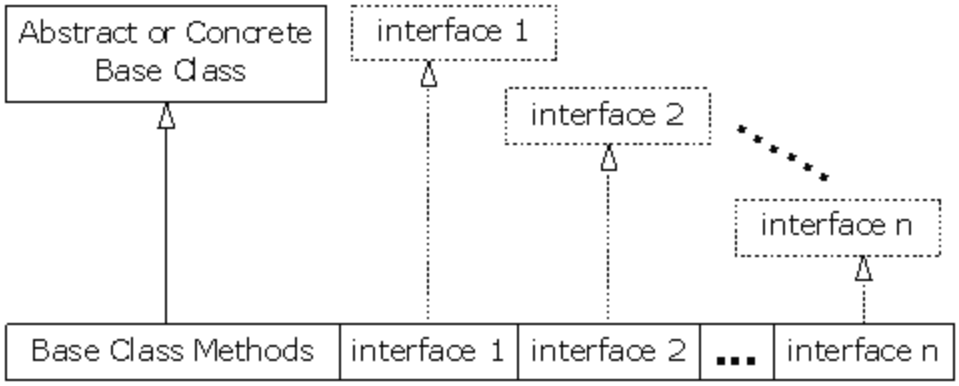
```
Washing
Spinning
Rinsing
Spinning
Constructor has done all the work!
```

Интерфейси

Интерфейс, групиращ само константи:

```
public interface Months {
  int
    JANUARY = 1, FEBRUARY = 2, MARCH = 3,
    APRIL = 4, MAY = 5, JUNE = 6, JULY = 7,
    AUGUST = 8, SEPTEMBER = 9, OCTOBER = 10,
    NOVEMBER = 11, DECEMBER = 12;
}
```

Наследяване на един клас и много интерфейси




```
interface CanFight {
  void fight();
}

interface CanSwim {
  void swim();
}

interface CanFly {
  void fly();
}

class ActionCharacter {
  public void fight() {}
}

class Hero extends ActionCharacter
                implements CanFight, CanSwim, CanFly {
  public void swim() {}
  public void fly() {}
}
```

```
public class Adventure {

  public static void t(CanFight x) { x.fight(); }
  public static void u(CanSwim x) { x.swim(); }
  public static void v(CanFly x) { x.fly(); }
  public static void w(ActionCharacter x) { x.fight(); }

  public static void main(String[] args) {
    Hero h = new Hero();
    t(h); // Treat it as a CanFight
    u(h); // Treat it as a CanSwim
    v(h); // Treat it as a CanFly
    w(h); // Treat it as an ActionCharacter
  }
}
```

Неоднозначност при наследяване на полета:

```
interface BaseColors {
      int RED = 1, GREEN = 2, BLUE = 4;
}

interface RainbowColors extends BaseColors {
      int YELLOW = 3, ORANGE = 5, INDIGO = 6, VIOLET = 7;
}

interface PrintColors extends BaseColors {
      int YELLOW = 8, CYAN = 16, MAGENTA = 32;
}

interface LotsOfColors extends RainbowColors, PrintColors {
      int FUCHSIA = 17, VERMILION = 43, CHARTREUSE = RED+90;
  //  int COLORLESS = YELLOW+1; // Ambiguous field
      int COLORLESS = PrintColor.YELLOW+1;
}
```

Колизия на имена при множествено наследяване на интерфейси:

```
  interface I1 { void f(); }

  interface I2 { int f(int i); }

  interface I3 { int f(); }
```

```
interface I4 { int f(); }

interface I5 extends I1, I2 {}

//! interface I6 extends I1, I3 {}

class C { public int f() { return 1; } }

class C2 implements I1, I2 {
        public void f() {}
        public int f(int i) { return 1; }
}

class C3 extends C implements I2 {
        public int f(int i) { return 1; } // overloaded
}

class C4 extends C implements I3 {
        // Identical, no problem:
        public int f() { return 1; } // overriden
}

class C5 extends C implements I3, I4 {
        public int f() {return 5;} // ???
        // try to ignore this method
}

// Methods differ only by return type:
//! class C5 extends C implements I1 {}

public class InterfaceCollision {
      public static void main(String args[]) {
            int i = new C5().f();
            System.out.println(i);
      }
}
```

Припокриване на методи от интерфейси – свежда се до уточняване на типа на връщаната стойност или ограничаване на множеството на пораждените от метода изключения

```
class BufferEmpty extends Exception {
    BufferEmpty() { super(); }
    BufferEmpty(String s) { super(s); }
}
```

```
class BufferException extends Exception {
      BufferException() { super(); }
      BufferException(String s) { super(s); }
}

public interface Buffer {
      char get() throws BufferEmpty, BufferException;
}

public interface InfiniteBuffer extends Buffer {
       char get() throws BufferException;       // override
}
```

Претоварване на методи – клас, реализиращ интерфейса *RealPointInterface* трябва да предвиди реализация на трите метода с име *move*.

```
interface PointInterface {
      void move(int dx, int dy);
}

interface RealPointInterface extends PointInterface {
      void move(float dx, float dy);
      void move(double dx, double dy);
}
```

Пример:

```
public interface Action {
  void act();
}

public class ActionOne implements Action{
  public void act() {
            System.out.println("Action one");
  }
}

public class ActionTwo implements Action {
  public void act() {
            System.out.println("Action two");
  }
}
```

```
public class Actions {

  Action a;

  void changeTo(Action a) { this.a=a;}

  void act() {a.act();}
}

public class Main {

  public static void main(String[] args) {
        Actions a = new Actions();
        a.act();  // ???
        a.changeTo(new ActionOne());
        a.act();
        a.changeTo(new ActionTwo());
        a.act();
  }
}
```

Задача 2: Да се напише клас за представяне на стек.

Задача 3: Да се въведе дума над азбуката { a, b, c } и да се провери дали е палиндром от вида $\alpha = \omega c \omega^{rev} / \omega \in \{a,b\}^+$.

Задача 4: Да се въведе последователност от цели числа завършваща с нула и да се изведат числата в обратен ред.