Programmering Fortsättning 2010

Övning 4

Joachim von Hacht

1 Arrayer

1. Kommer koden nedan att kompilera? Kommer det att fungera runtime?

```
package inherit.array;
public class A {
    public void doA(){
        System.out.println(this.getClass() + " doing A");
    }
}

package inherit.array;
public class B extends A {
    public void doB(){
        System.out.println( this.getClass() + " doing B");
    }
}

package inherit.array;
public class Main {
    public static void main(String args[]) {
       B[] b = new B[]{ new B(), new B(), new B() };
       upDate( b );
       b[1].doB();
    }
    public static void upDate( A[] a ){
        a[0] = null;
        a[1] = new A();
        a[2] = new B();
    }
}
```

1p

2 "Sköra basklass"-problemet

1. (Fragile baseclass problem, FBC) Vi har skapat en subklass MonitorableStack med MyStack som basklass;

```
package inherit.fbc;
import java.util.ArrayList;
/**
 * Simple stack.
 * @inv this.pop(this.push(s)) == s && this.stackPointer >= 0 &&
 *      this.size() == this.stackPointer
 */
class MyStack extends ArrayList<Object> {
    private static final long serialVersionUID = 1L;
    private int stackPointer = 0;
    public void push(Object article) {
        add(stackPointer, article);
        stackPointer++;
    }
    public Object pop() {
        stackPointer--;
        return remove(stackPointer);
    }
    public void pushMany(Object[] o) {
        for (int i = 0; i < o.length; ++i)
            push(o[i]);
    }
}
```

```
package inherit.fbc;
/**
 *  A stack that stores the maximum size over its entire lifetime
 *  @inv highWaterMark >= currentSize (&& MyStack invariants)
 */
class MonitorableStack extends MyStack {
    private static final long serialVersionUID = 1L;
    private int highWaterMark = 0;
    private int currentSize;
    public void push(Object article) {
        currentSize++;
        if (currentSize > highWaterMark) {
            highWaterMark = currentSize;
        }
        super.push(article);
```

```
    }
    public Object pop() {
        currentSize--;
        return super.pop();
    }
    public int maximumSizeSoFar() {
        return highWaterMark;
    }
}
```

Vi vill kunna göra följande t.ex.;

```
package inherit.fbc;
public class Main {
    public static void main(String[] args) {
        Object[] strs = new Object[] {"a", "b", "c", "d", "e"};
        MonitorableStack m = new MonitorableStack();
        m.pushMany(strs);
        System.out.println(m.maximumSizeSoFar());// 5 ok
        m.push("f");
        System.out.println(m.maximumSizeSoFar()); // 6 ok
    }
}
```

Visa att man kan knäcka MonitorableStack genom att byta representation och ändra i metoden pushMany i basklassen (s.k. tillåtna förändringar som inte skall påverka kontraktet). 2p

3 Arv och överlagring

1. Vad skrivs ut? Vad är det som händer??

```
package inherit.contra;
public class Main {
    public static void main(String[] args) {
        Car c1 = new Car(123,1);
        Car c2 = new Car(123,2);
        Vehicle v1 = c1;
        Vehicle v2 = new Vehicle(123);
        if( v1.equals(c2)){
            System.out.println("v1 eq c2");
        }
        if( !v1.equals(v2)){
```

```
            System.out.println("v1 NOT eq v2");
        }
    }
}

package inherit.contra;
public class Vehicle {
    protected int licNumb;
    public Vehicle(int licNumb) {
        this.licNumb = licNumb;
    }
    public boolean equals(Vehicle v) {
        if (this == v) {
            return true;
        }else if (v == null) {
            return false;
        }else if (getClass() != v.getClass()) {
            return false;
        }
        final Vehicle other = (Vehicle) v;
        if (licNumb != other.licNumb){
            return false;
        }
        return true;
    }
}

package inherit.contra;
public class Car extends Vehicle {
    private int numbSeats;
    public Car(int licNumb, int numbSeats) {
        super(licNumb);
        this.numbSeats = numbSeats;
    }
    public boolean equals(Car c) {
        if (this == c) {
            return true;
        } else if (!super.equals(c)) {
            return false;
        } else if (getClass() != c.getClass()) {
            return false;
        }
        final Car other = (Car) c;
        if (numbSeats != other.numbSeats){
```

```
            return false;
        }
        return true;
    }
}
```

2p

4 Inre klasser

1. Vilka rader kommer inte att kompilera? Om man kommenterar bort dessa och kör vad kommer att skrivas ut?

```
package inner;

//Thanks to Wayne Pollock, Tampa Florida USA.
class Outer {
  int num = 1;
  static int staticNum = 1;
  class Inner {
    int num = 2;
    void aMethod() {
      System.out.println("num=" + num);
      System.out.println("this.num=" + this.num);
      System.out.println("Inner.this.num=" + Inner.this.num);
      System.out.println("Outer.this.num=" + Outer.this.num);
      System.out.println("staticNum=" + staticNum);
      System.out.println("this.staticNum=" + this.staticNum);
      System.out.println("Inner.this.staticNum=" + Inner.this.staticNum);
      System.out.println("Outer.this.staticNum=" + Outer.this.staticNum);//Waring
      System.out.println("Outer.staticNum=" + Outer.staticNum);
    }
  }

  static class Nested {  //Note static
    int num = 3;
    void aMethod() {
      System.out.println("num=" + num);
      System.out.println("this.num=" + this.num);
      System.out.println("Nested.this.num=" + Nested.this.num);
      System.out.println("Outer.this.num=" + Outer.this.num);
      System.out.println("staticNum=" + staticNum);
      System.out.println("Outer.staticNum=" + Outer.staticNum);
      System.out.println("Nested.staticNum=" + Nested.staticNum);
      System.out.println("this.staticNum=" + this.staticNum);
```

```
        System.out.println("Nested.this.staticNum=" + Nested.this.staticNum);
        System.out.println("Outer.this.staticNum=" + Outer.this.staticNum);
      }
    }
}
```

2p

5 Generiska klasser

1. Gör om klassen nedan så att den blir generisk. Implementera därefter en (generisk) iterator, som en inre klass. Att använda casts är i detta fall tillåtet (vi kan inte göra new E[..]). Vad skall hända om man modifierar klassen under tiden man genomlöper den med iteratorn?

```
package iterator;
import java.util.Arrays;
public class DataStructure {
   // Bad but used for now
   private Object[] elementData;
   // Actual number of elements
   private int size = 0;
   // Max number of elements
   private int capacity = 4;
   public DataStructure() {
      elementData = new Object[capacity];
   }
   public boolean add(Object e) {
      if (size == capacity - 1) {
         capacity *= 2;
         elementData = Arrays.copyOf(elementData, capacity);
      }
      elementData[size++] = e;
      return true;
   }
   public void clear() {
      for (int i = 0; i < size; i++) {
         elementData[i] = null; // Let gc do its work
      }
      capacity = 4;
      elementData = new Object[capacity];
      size = 0;
   }
   public int size() {
      return size;
```

```
    }
    /*
    @Override
    public Iterator iterator() {
      // TODO
    }
     */
}
```

3p