Klausur der LV Programmierung II im WS 06/07

Name:

Matr. Nr.

Punkte: 58 Note 1.3

Datum: 8.2.07

Dies ist mein 3. Prüfungsversuch zu dieser Lehrveranstaltung: Ja o Nein x

Achtung, bearbeiten Sie die Klausur keinesfalls mit Bleistiften oder unter Verwendung roter Farbe. Geben Sie auf jeder Seite Ihren Namen und Ihre Matrikel-Nr. an. Es werden nur leicht lesbare Klausuren bewertet.

---

(1) (max. 15 Punkte)

Was gibt das folgende Programm auf die Konsole aus?

```
public class BytePlay {

    public static byte modify1(byte b) {
        int b1 = (b & 0xF0) >> 4;
        int b2 = (b & 0x0F) << 4;
        return (byte)(b1|b2);
    }

    public static byte modify2(byte b) {
        return (byte)(b^(2<<6));
    }

    public static int create() {
        return (2|8|(7&4));
    }

    public static String toBitString(byte b) {
            byte mask = 1;
            char[] ca = new char[8];
            for (int i= 0; i< 8;i++){
                ca[7-i] = (b & mask) == 0 ? '0' : '1';
                mask <<= 1;
            }
            return new String(ca);
    }

    public static void main(String[] args) {
        byte b1 = 5;
        System.out.println(toBitString(modify1(b1)));
        System.out.println(toBitString(modify1(modify1(b1))));
        byte b2 = -22;
        System.out.println(modify2(b2));
        System.out.println(modify2(modify2(b2)));
        System.out.println(create());
    }
}
```

Ausgaben:

→ 01010000 ✓

→ 00000101 ✓

→ 22 ↑ 106

→ -22 ✓

→ 14 ✓

Taste drücken um Programm zu beenden...

12 P

(2) (max. 15 Punkte)

Analysieren Sie den folgenden Code-Ausschnitt, und beantworten Sie die folgenden Fragen so konkret und genau wie möglich:

```
item.addActionListener (new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            doXY();
        }
});
```

a) Was ist *item* genau? item ist die Instanz einer Klasse, auf die ein ActionListener angewendet werden kann, das ein Ereignis auslöst. Von der Namensgebung her wahrscheinlich ein Menüeintrag.

b) Was ist *new ActionListener() {public void actionPerformed (ActionEvent e) {doXY();}}* ?
Es ist die Implementierung einer Anonymen innerenKlasse mit der Implementierung der dazugehörigen Methode actionPerformed, welche die Methode doXY() bei Ereignis auslöst.

c) Was ist *{public void actionPerformed (ActionEvent e) {doXY();}}* ?
Im Prinzip was ich im zweiten Teil dazu sagte.
Es ist eine Methode der Klasse ActionListener. Sie wird bei aufgetretenem Ereignis ausgelöst und bekommt das ActionEvent übergeben. Im Beispiel wird die Prozedur doXY(); ausgelöst. ActionEvent e kann auf die Eigenschaften abgefragt werden.

15P

(3) (max. 15 Punkte)

In der Klasse *HardWork* befindet sich eine statische Methode *beBusy*, deren Ausführung verhältnismäßig lange dauert. Delegieren Sie die Ausführung dieser Methode an einen separaten Thread der Klasse MainClass (also nicht an den main-Thread).

```
class HardWork {

    public static void beBusy(int howBusy) {
        double d = 0;
        for(int j = 1; j < howBusy; j++)
            for(int i = 1; i < 100000; i++)
                d = d + (Math.PI + Math.E) / (double)i;
        System.out.println("uff!");
    }
}

public class MainClass {
    public static void main(String[] args){
    }
}
```

1. Thread

neuer Thread

```
javax.swing.SwingUtilities.invokeLater(new Runnable(){
    public void run(){
        beBusy(1000);
    }});
```

HardWork.

13P

suboptimal weil mehr Threads erzeugt werden (durch Swing) als notwendig.

(4) (max. 15 Punkte)

Gegeben sei folgendes Interface:

```
interface Priority
        public int getPriority();
}
```

a) Implementieren Sie in der Klasse *MyUtils* eine generische Methode mit dem Namen *firstOne* mit folgenden Eigenschaften:

Input Parameter:	2 Referenzen von Typ *Priority*
Return Value:	Referenz des Parameters mit der größten Priorität (oder wenn die Priorität übereinstimmt, die Referenz des 1. Parameters).

b) Fügen Sie in die Main-Methode einen Testaufruf ein.

c) Begründen Sie, warum eine Methode mit der Schnittstelle

public Priority firstOne(Priority p1, Priority p2)

nicht geeignet wäre.

```
class MyUtils {

}

class Candidate implements Priority {
    private String name;
    private int prio;

    public Candidate(String name, int prio) {
        this.name = name;
        this.prio = prio;
    }

    public int getPriority() {
        return prio;
    }

    public String toString() {
        return name + ": " + prio;
    }
}

public class Test {

    public static void main(String[] args) {
        Candidate c1 = new Candidate("Max", 15);
        Candidate c2 = new Candidate("Moritz", 15);
        // test here
    }
}
```

static public <T extends Priority> T firstOne(T one, T two) {
}*

* if(one.getPriority() >= two.getPriority())
return one;
return two;

System.out.println(firstOne(c1,c2)); → Console "Max: 15"
System.out.println(firstOne(c2,c1)); → Console "Moritz: 15"

10 P

c) Sie wäre ungeeignet, da ich mich auf den Typen Priority festlege. Durch die Verwendung eines Generischen Methode gewinne ich eine höhere Freiheit. Ich kann alle Klassen, die eine Methode/Interface getPriority()/Priority implementieren vergleichen und wäre nicht auf das Interface Priority fixiert. f

(5) (max. 10 Punkte)

Analysieren Sie das folgende Programm, und beschreiben Sie exakt, was es tut.

```
import java.io.*;

public class XClass implements Serializable {

    Object o;

    public XClass(Object o) {
        this.o = o;
    }

    public static void main(String[] args) throws IOException{
        XClass n = new XClass(new Object());
        File file = new File("test.ser");
        FileOutputStream outFile = new FileOutputStream(file);
        ObjectOutputStream outStream = new ObjectOutputStream(outFile);
        outStream.writeObject(n);
        outStream.close();
        outFile.close();
    }
}
```

a)

b)

1) 2) 3) 4) 5) 6) 7)

Ich dachte wir dürfen nicht mit dem Vorschlägramms anfangen *motz*! Das Lernen des Exceptions hat anderes wertvolles Wissen verdrängt :-

Richtig, stilistisch ist das schlecht. Ich wollte das Programm möglichst kurz haben.

1) Holen einer Instanz n von Typ XClass mit anonymen Object. Object(); Object mit Standardkonstruktor.

2) Holen einer Instanz von File file. "test.ser" wird angelegt falls nicht vorhanden.
! Warnung vor NullPointerExceptions von Klasse File.

3) Holen einer Instanz von Typ FileOutputStream auf dem soeben File file.
! Warnung vor FileNotFoundException.

4) Holen einer Instanz von ObjectOutputStream auf dem FileOutputStream.
! Warnung vor IOException!
Voraussetzungen zum Serialisieren (sehr einfach!) geschaffen, da:

a) • die Klasse XClass das Interface Serializable implementiert

b) • die Klasse Object serialisierbar ist ist sie nicht

c) • keine nicht serialisierbaren Objekte „weg geschrieben" werden sollen

5) Schreiben des Objektes n der Klasse XClass mit der Methode writeObject von ObjectOutputStream

6) Schließen des ObjectOutputStream (geschieht auch nach Beenden des Programms)

7) Schließen des FileOutputStreams (geschieht auch nach Beendigung des Programms)

Zusammengefasst = Es wurde ein sinnfreies Objekt, welches ein Objekt vom Typ Object enthält in ein (neu erstelltes) File "test.ser" gespeichert und anschließend das Programm beendet. Jetzt könnte man es wieder laden, wenn man will.... nee! Es gibt eine Exc. ü

8P

1)

b	0101
FO_H &	1111 0000

$b \& FO_{Hex}$ 0000 0000 >> 4

$b \& FO_H >> 4$ 0000 0000 = b1

b	0101
OF_{Hex}	0000 1111

$b \& F_{Hex}$ 0000 0101 << 4

$b \& F_{Hex} << 4$ 0101 0000 = b2
0000 0000 = b1

b1 | b2 0101 0000 = 80_{Dec}

64 16

char ca |0|1|0|1|0|0|0|0|
←

} toBitString(modify(b1))

b	0101
FO_{Hex}	1111 0000

$b \& FO_{Hex}$ 0000 0000 = b1
0101 0000 = b2
⋮
b1 | b2 0101 0000 = neues b

neu:

b	0101 0000
FO_{Hex}	1111 0000
&	0101 0000
>> 4	0000 0101 = b1

OF_{Hex}

b	0101 0000
OF_{Hex}	0000 1111
&	0000 0000
<< 4	0000 0000 = b2

b1 | b2 0000 0101 = 5_{Dez}

Char ca |▮|0|0|0|0|0|1|0|1|
←

} toBitString(modify1(modify(b1)))

Signum

b2 = -22 = (1)0010110

b = 10010110

2<<6|2 = 00000010
<<6 = 10000000
b×02 = 00010110

} modify1(b2),

Dekrement des Vorzeichen. s.B.

} modify1(modify2(b2))

7 0111
4 0100
& And 0100 = 4
8 1000
| Or 1100 = 12
2 0010
| OR 1110 = 14

} create()