

| | |
|---|--|
| FACH: CAD-Projekt | NAME: |
| DATUM: Donnerstag, 18.12.2008 | |
| ZEIT: 15:45 – 17:15 | SEMESTER: |
| PRÜFER: Prof. Dr. Wolfgang Erben | STUDIENGANG: <input type="checkbox"/> Diplom Mathematik <input type="checkbox"/> Bachelor Mathematik |

- ANLAGEN:** • Quelltext einiger Klassen (**bereits vorab ausgegeben**).
- HILFSMITTEL:** • Handschriftliche Ergänzungen auf der vorab ausgegebenen Anlage

- UNBEDINGT BEACHTEN:**
- Tragen Sie **jetzt gleich** auf dieser Seite Ihren **Name** und ihr **Semester** ein und kreuzen Sie Ihren **Studiengang** an. Bearbeitungen ohne diese vorherige Kennzeichnung sind ungültig.
 - Führen Sie die Bearbeitung direkt auf diesen Aufgabenblättern in den dafür vorgesehenen Rahmen aus. Der dort vorhandene Platz sollte ausreichen.

- HINWEISE ZUM QUELLTEXT:**
- Bei dem angegebenen Quellcode handelt es sich um Auszüge, welche sinnvoll ergänzt zu denken sind. Insbesondere trifft dies auf das Importieren benötigter Klassen in anderen Packages zu.
 - `Intervall` und `VereinigungVonIntervallen` sind zwei Klassen, welche den Anforderungen der (zweiten) Studienarbeit genügen. Insbesondere sind beide Klassen kanonisch.
 - An vielen Stellen ist der Quelltext absichtlich schlecht. Insbesondere werden häufig nichtssagende Namen verwendet.

- EMPFEHLUNGEN ZUR BEARBEITUNG:**
- Lesen Sie die Aufgaben und Aufgabenteile bitte in der vorgegebenen Reihenfolge durch. An manchen Stellen ist dies für das Verständnis der Aufgabenstellung wichtig.
 - Einzelne Bearbeitungen können aber zurückgestellt werden. Die vollständige Lösung einer Aufgabe ist für die folgenden Aufgaben nicht erforderlich.

Aufgabe 1: (12 Punkte)

In dieser Aufgabe geht es um allgemeine Zusammenhänge.

a) Welche der folgenden Aussagen sind richtig?

| | ja | nein |
|--|----|------|
| Jede eigene Klasse besitzt eine Basisklasse. | ✓ | |
| Im Konstruktor einer Klasse wird immer (explizit oder implizit) ein Konstruktor der Basisklasse gerufen. | ✓ | |
| Jede Klasse besitzt eine öffentliche <code>equals</code> Methode. | ✓ | |
| Konstruktoren können überladen werden. | ✓ | |
| Konstruktoren können überschrieben werden. | | ✓ |

b) Welche der folgenden Aussagen sind richtig?

| | ja | nein |
|--|----|------|
| Eine Klasse kann höchstens ein <code>interface</code> implementieren. | | ✓ |
| Ein <code>interface</code> kann nicht als Rückgabotyp einer Methode verwendet werden. | | ✓ |
| Ein <code>interface</code> muss mindestens eine Methode enthalten. | | ✓ |
| Ein <code>interface</code> kann nur öffentliche Methoden haben. | ✓ | |
| Die Methoden eines <code>interface</code> müssen explizit als <code>public</code> gekennzeichnet werden. | | ✓ |

c) Welche der folgenden Aussagen sind richtig?

| | ja | nein |
|--|----|------|
| Eine abstrakte Klasse kann keinen Konstruktor besitzen. | | ✓ |
| Eine abstrakte Klasse besitzt mindestens eine abstrakte Methode. | | ✓ |
| Eine abstrakte Klasse darf nur abstrakte Methoden besitzen. | | ✓ |
| Eine Klasse mit einer abstrakten Methode muss explizit als abstrakt deklariert werden. | ✓ | |
| Von einer abstrakten Klasse kann nicht abgeleitet werden. | | ✓ |

d) Welche der folgenden Empfehlungen sollten möglichst befolgt werden?

| | ja | nein |
|--|----|------|
| In einem Konstruktor sollten keine virtuellen Methoden gerufen werden. | ✓ | |
| <code>hashCode</code> sollte für gleiche Objekte den gleichen Wert liefern. | ✓ | |
| <code>hashCode</code> muss für verschiedene Objekte verschiedene Werte liefern. | | ✓ |
| <code>equals</code> muss eine Äquivalenzrelation sein. | ✓ | |
| Bei Klassen mit <code>double</code> -Werten sollte <code>equals</code> die Gleichheit mit Toleranz überprüfen. | | ✓ |

Aufgabe 2: (12 Punkte)

In dieser Aufgabe geht es um Design-Entscheidungen bei den Klassen `Intervall` und `VereinigungVonIntervallen` aus den Studienarbeiten.

a) In beiden Klassen wird die Methode `equals` überschrieben. Warum?

Die von Object geerbte Methode vergleicht nur die Adressen.
Benötigt wird aber ein Vergleich der abstrakten Werte.

Mit `equals` wird auch die Methode

`hashCode`

überschrieben.

b) Die Methode `clone` wird nur in `VereinigungVonIntervallen` überschrieben. Warum nicht in der Klasse `Intervall`?

Die Klasse `Intervall` ist unveränderbar.
Mehrfachnutzung von Objekten ist deshalb ungefährlich.
Kopieren ist unnötig und sollte vermieden werden.

c) Die beiden Klassen `Intervall` und `VereinigungVonIntervallen` könnten ein gemeinsames `interface`, etwa `Menge` besitzen. Was spricht dagegen?

`Intervall` soll unveränderbar sein,
`VereinigungVonIntervallen` hingegen nicht.
Viele Methoden, etwa für Schnitt und Vereinigung
besitzen unterschiedliche Logik und unterschiedliche
Signatur.

Aufgabe 3: (27 Punkte)

In dieser Aufgabe geht es um die Parameter-Übergabe. Aus den Studienarbeiten wird die Klasse `Intervall` und aus dem **Anhang** die Klasse `Tool` verwendet.

```
public class Murks {
    private Intervall i = Tool.gibMir();
    public final Murks tuWasMit(Murks a) {
        if (i.istDisjunktZu(a.i)) a = this;
        i = i.untereHaelfte();
        a.i = a.i.obereHaelfte();
        if (i.equals(a.i)) return this;
        return new Murks();
    }
}
```

Der abstrakte Wert eines Objektes dieser Klasse ist der des enthaltenen Intervalls. Nach den drei Initialisierungen

```
Murks a = new Murks();
Murks b = new Murks();
Murks c = a;
```

haben die Objekte `a`, `b` und `c` die (abstrakten) Werte

| | | | | | |
|---|-----------------------|---|-----------------------|---|-----------------------|
| a | <code>[-1, 15]</code> | b | <code>[14, 30]</code> | c | <code>[-1, 15]</code> |
|---|-----------------------|---|-----------------------|---|-----------------------|

a) Wie verändern sich diese Werte durch folgende Anweisung?

```
c = a.tuWasMit(b);
```

| | | | | | |
|---|----------------------|---|-----------------------|---|-----------------------|
| a | <code>[-1, 7]</code> | b | <code>[22, 30]</code> | c | <code>[29, 45]</code> |
|---|----------------------|---|-----------------------|---|-----------------------|

b) Wie verändern sich diese neuen Werte durch folgende Anweisung?

```
b = c.tuWasMit(a);
```

| | | | | | |
|---|----------------------|---|-----------------------|---|-----------------------|
| a | <code>[-1, 7]</code> | b | <code>[33, 37]</code> | c | <code>[33, 37]</code> |
|---|----------------------|---|-----------------------|---|-----------------------|

c) Damit erfolge nachstehender Aufruf. Welchen Wert haben `a`, `b` und `c` dann?

```
a = c.tuWasMit(b);
```

| | | | | | |
|---|-----------------------|---|-----------------------|---|-----------------------|
| a | <code>[34, 35]</code> | b | <code>[34, 35]</code> | c | <code>[34, 35]</code> |
|---|-----------------------|---|-----------------------|---|-----------------------|

Aufgabe 4: (21 Punkte)

In dieser Aufgabe geht es um das Kopieren von Objekten der in der **Anlage** definierten Typen. Eine JUnit-Testklasse enthält die Hilfs-Methode

```
private void kopiere(Pool p) {
    Pool q = p.clone();
    assertTrue(p.equals(q)); // A
    assertTrue(q.equals(p)); // B
    assertFalse(p == q); // C
    q.gibMir();
    assertFalse(p.equals(q)); // D
    assertFalse(q.equals(p)); // E
    assertTrue(p == q); // F
}
```

Verschiedene – in der ersten Spalte nachfolgender Tabelle angegebene – Testmethoden bestehen jeweils aus einem einzigen Aufruf dieser Hilfs-Methode. Keine dieser Testmethoden läuft fehlerfrei ab. Geben Sie die Stelle (A, B, ..., F) an, bei welcher der Fehler auftritt:

| | |
|--|----------|
| <pre>public void testeKopierenToolPool() { kopiere(new ToolPool()); }</pre> | C |
| <pre>public void testeKopierenCoolPool() { kopiere(new CoolPool()); }</pre> | F |
| <pre>public void testeKopierenBoolPool() { kopiere(new BoolPool()); }</pre> | F |
| <pre>public void testeKopierenFoolPool() { kopiere(new FoolPool()); }</pre> | B |
| <pre>public void testeKopierenToolPoolPool() { kopiere(new PoolPool(new ToolPool())); }</pre> | F |
| <pre>public void testeKopierenFoolPoolPool() { kopiere(new PoolPool(new FoolPool())); }</pre> | A |
| <pre>public void testeKopierenFoolPoolPoolPool() { kopiere(new PoolPool(new PoolPool(new FoolPool()))); }</pre> | B |

Aufgabe 5: (18 Punkte)

In dieser Aufgabe geht es um die Verwendung der Klassen aus der **Anlage**.

a) Kreuzen Sie jeweils an, ob die angegebene Verwendung korrekt ist oder aber zu einem Compile-Fehler, einer Warnung (inkl. Checkstyle) oder zu einer Exception führt:

| Verwendung | C.-F. | War. | Exc. | OK |
|---|-------|------|------|----|
| Pool p = new ToolPool(); Intervall s = (Intervall) p.gibMir(); | | | | ✓ |
| Pool p = new FoolPool(); Intervall s = (Intervall) p.gibMir(); | | | ✓ | |
| FoolPool p = new FoolPool(); Intervall s = (Intervall) p.gibMir(); | ✓ | | | |
| Pool p = new CoolPool(); Intervall s = p.gibMir(); | ✓ | | | |
| CoolPool p = new BoolPool(); Intervall s = p.gibMir(); | | | | ✓ |
| Tool p = new Tool(); Intervall s = p.gibMir(); | | ✓ | | |

b) Die Anweisungen in der linken Tabellenspalte werden hintereinander ausgeführt. Geben Sie in der rechten Spalte den Hash-Code von `p` nach der Ausführung des jeweiligen Statements an:

| | <code>p.hashCode()</code> |
|---------------------------------------|---------------------------|
| <code>Pool p = new BoolPool();</code> | 1 |
| <code>p.gibMir();</code> | 3 |
| <code>p.nimmDir(2);</code> | 8 |