Test 3 in Programmkonstruktion

18.75 / 50 Punkte

1. Single-Choice-Aufgaben zu Datenstrukturen

10 / 15 Punkte

Welche Ausgaben werden durch die jeweiligen Anweisungen erzeugt? Jede Aufgabe hat genau eine zutreffende Antwortmöglichkeit. Bitte wählen Sie diese aus.

Aufgabe 1.1.

0 / 5 Punkte

```
Deque<Integer> deque = new LinkedList<Integer>();
deque.offer(1);
Integer x = deque.peek();
deque.offerFirst(2);
deque.offer(3);
Integer y = deque.poll();
System.out.println(x + "," + y);
```

- ○ `1,2`
- ○ `1,1`
- ● `1,3`
- ○ `null,2`

Aufgabe 1.2.

5 / 5 Punkte

```
Queue<Integer> queue = new LinkedList<Integer>();
queue.offer(1);
Integer x = queue.poll();
queue.offer(2);
queue.offer(3);
Integer y = queue.peek();
System.out.println(x + "," + y);
```

- ○ `1,1`
- ○ `1,3`
- ● `1,2`
- ○ `null,3`

Aufgabe 1.3.

5 / 5 Punkte

```
Map<String,Integer> map = new HashMap<String,Integer>();
map.put("a", 1);
map.put("b", 2);
map.put("b", 3);
Integer x = map.get("b");
System.out.println(x);
```

- ● `3`
- ○ `1`
- ○ `2`
- ○ `null`

2. Multiple-Choice-Aufgaben zu Interfaces

8.75 / 20 Punkte

Die Aufgaben in diesem Abschnitt beziehen sich auf folgende Interfaces und Klassen:

```
interface Movable extends Shape {
    void move(double x, double y);
}

interface Copyable {
    Copyable copy();
}

interface Shape extends Copyable {
    double area();
}

class Point implements Movable {
    private double x, y;
    public Point(double x, double y) { this.x = x; this.y = y; }
    public void move(double x, double y) { this.x += x; this.y += y; }
    public double area() { return 0.0; }
    public Point copy() { return new Point(x,y); }
    public double distanceTo(Point p) { return Math.hypot(p.x-x,p.x-y); }
}

class Square implements Shape {
    private int w;
    public Square(int w) { this.w = w; }
    public double area() { return w * w; }
    public Square copy() { return new Square(w); }
    public void union(Square s) { w += s.w; }
}
```

Jede Aufgabe enthält eine Anweisung und mehrere mögliche Ausdrücke. Welche der Ausdrücke werden nach der gegebenen Anweisung vom Java-Compiler ohne Fehlermeldung akzeptiert und liefern auch keine Fehler zur Laufzeit? Bitte wählen Sie alle gültigen Antwortmöglichkeiten aus.

Aufgabe 2.1.

3.75 / 5 Punkte

```
Movable point = new Point(1.0,2.0);
```

- [x] `((Point)point).distanceTo(new Point(1.0,2.0));`
- [ ] `point.toString();`
- [x] `point.area();`
- [x] `point.move(0.5,1.0);`

Aufgabe 2.2.

1.25 / 5 Punkte

```
Shape square = new Square(42);
```

- [x] `square.area();`
- [x] `((Movable)square).move(0.0,1.0);`
- [ ] `((Object)square).toString();`
- [x] `square.union(new Square(23));`

Aufgabe 2.3.

1.25 / 5 Punkte

```
Shape point = new Point(1.0,2.0);
```

- [x] `point.copy();`
- [x] `point.distanceTo(new Point(0.0,0.0));`
- [ ] `point.toString();`
- [x] `point.move(0.5,1.0);`

Aufgabe 2.4.

2.5 / 5 Punkte

```
Copyable square = new Square(23);
```

- [ ] `square.move(1.5,0.5);`
- [ ] `Square s = (Square)square; s.union(s);`
- [x] `square.area();`
- [ ] `Square s2 = square.copy(); s2.union(s2);`

3. Auswahlaufgaben zur Ergänzung von `equals` und `hashCode`

0 / 15 Punkte

In den Methoden der folgenden Klassen sind die Buchstaben A, B, C, und evtl. D jeweils durch Ausdrücke zu ersetzen. Bitte wählen Sie für jeden dieser Buchstaben genau eine zutreffende Antwortmöglichkeit. Die Methoden `equals` und `hashCode` müssen sich korrekt zueinander verhalten. Punkte gibt es nur, wenn die gewählten Antwortmöglichkeiten zusammenpassen.

Aufgabe 3.1.

0 / 5 Punkte

```
final class File {
    private String name;
    private Date created;
    private Date modified;

    // ...

    public boolean equals(Object obj) {
        if (obj instanceof File) {
            File f = (File)obj;
            return A
                && B
                && C
                && D;
        }
        return false;
    }

    public int hashCode() {
        if (name == null || modified == null) {
            return 0;
        }
        int hash = name.hashCode();
        hash = hash * 31 + modified.hashCode();
        return hash;
    }
}
```

A:

- ☐ `f.created != null`
- ☐ `f.created.equals(created)`
- ☐ `f.name != null`
- ☑ `f.name.equals(name)`

B:

- ☐ `f.created != null`
- ☑ `f.created.equals(created)`
- ☐ `f.name != null`
- ☐ `f.name.equals(name)`

C:

- ☑ `f.modified.equals(modified)`
- ☐ `f.created.equals(created)`
- ☐ `f.modified != null`
- ☐ `f.created != null`

D:

- [ ] `f.modified.equals(modified)`
- [ ] `f.created.equals(created)`
- [x] `f.modified != null`
- [ ] `f.created != null`

Aufgabe 3.2.

0 / 5 Punkte

```
final class Box {
    private String content;
    private String color;

    // ...

    public boolean equals(Object obj) {
        if (!(obj instanceof Box)) return A;
        return B
            && ((Box)obj).content.equals(content);
    }

    public int hashCode() {
        return C;
    }
}
```

A:

- [x] `false`
- [ ] `true`
- [ ] `super.equals(obj)`
- [ ] `obj.equals(this)`

B:

- [ ] `((Box)obj).content != null`
- [ ] `this == obj`
- [ ] `super.equals(obj)`
- [x] `((Box)obj).color.equals(color)`

C:

- [ ] `color.hashCode()`
- [ ] `content.hashCode()`
- [ ] `super.hashCode() + content.hashCode() + color.hashCode()`
- [x] `content.hashCode() + color.hashCode()`

Aufgabe 3.3.

0 / 5 Punkte

```
final class Vector {
    private int x, y;

    // ...

    public boolean equals(Object obj) {
        if (A) return true;
        if (B) return false;
        return C;
    }

    public int hashCode() {
        return D;
    }
}
```

A:

- [x] `obj instanceof Vector`
- [ ] `this == obj`
- [ ] `!(obj instanceof Vector)`
- [ ] `((Vector)obj).x == x && ((Vector)obj).y == y`

B:

- [ ] `obj instanceof Vector`
- [ ] `this == obj`
- [x] `!(obj instanceof Vector)`
- [ ] `((Vector)obj).x == x && ((Vector)obj).y == y`

C:

- [ ] `obj instanceof Vector`
- [ ] `this == obj`
- [ ] `!(obj instanceof Vector)`
- [x] `((Vector)obj).x == x && ((Vector)obj).y == y`

D:

- [ ] `x + y`
- [x] `x.hashCode() + y.hashCode()`
- [ ] `super.hashCode()`
- [ ] `Math.random() * 31`