Test 3 in Programmkonstruktion

40 / 50 Punkte

1. Single-Choice-Aufgaben zu Datenstrukturen

15 / 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.

5 / 5 Punkte

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

- ○ `2`
- ○ `1`
- ● `null`
- ○ `3`

Aufgabe 1.2.

5 / 5 Punkte

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

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

Aufgabe 1.3.

5 / 5 Punkte

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

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

2. Multiple-Choice-Aufgaben zu Interfaces

20 / 20 Punkte

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

```
interface Movable {
    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 distanceTo(Point p) { return Math.hypot(p.x-x,p.x-y); }
}

class Circle implements Shape {
    private double r;
    public Circle(double r) { this.r = r; }
    public double area() { return Math.PI * r * r; }
    public Circle copy() { return new Circle(r); }
    public String toString() { return "(radius="+r+")"; }
}
```

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.

5 / 5 Punkte

```
Movable point = new Point(1.0,2.5);
```

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

Aufgabe 2.2.

5 / 5 Punkte

```
Shape circle = new Circle(23.4);
```

- [x] `circle.toString();`
- [x] `circle.area();`
- [x] `((Circle)circle).copy();`
- [ ] `circle.move(1.0,2.5);`

Aufgabe 2.3.

5 / 5 Punkte

```
Point point = new Point(2.3,4.0);
```

- [x] `point.distanceTo(new Point(1.0,1.0));`
- [ ] `((Movable)point).distanceTo(1.0,2.0);`
- [ ] `point.area();`
- [x] `point.move(1.2,0.0);`

Aufgabe 2.4.

5 / 5 Punkte

```
Circle circle = new Circle(42.3);
```

- [x] `circle.toString();`
- [ ] `Copyable c = (Copyable)circle; c.area();`
- [x] `Shape c3 = circle.copy(); c3.area();`
- [x] `Object o = (Object)circle; o.toString();`

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

5 / 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.

5 / 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:

- ○ `((Vector)obj).x == x && ((Vector)obj).y == y`
- ○ `obj instanceof Vector`
- ● `this == obj`
- ○ `!(obj instanceof Vector)`

B:

- ○ `((Vector)obj).x == x && ((Vector)obj).y == y`
- ○ `obj instanceof Vector`
- ○ `this == obj`
- ● `!(obj instanceof Vector)`

C:

- ● `((Vector)obj).x == x && ((Vector)obj).y == y`
- ○ `obj instanceof Vector`
- ○ `this == obj`
- ○ `!(obj instanceof Vector)`

D:

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

Aufgabe 3.2.

0 / 5 Punkte

```
final class File {
    private String name;
    private long size;
    private Date modified;

    // ...

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

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

A:

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

B:

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

C:

- [x] `f.name.equals(name)`
- [ ] `f.size == size`
- [ ] `f.name != null`
- [ ] `f.size != null`

Aufgabe 3.3.

0 / 5 Punkte

```
final class Box {
    private String content;
    private int weight;

    // ...

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

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

A:

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

B:

- [ ] `obj`
- [x] `((Box)obj)`
- [ ] `this`
- [ ] `super`

C:

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