Test 3 in Programmkonstruktion

35 / 40 Punkte

Alle Aufgaben beziehen sich auf Java.

1. Multiple-Choice-Aufgaben zu Interfaces

23.75 / 25 Punkte

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

```
interface Copyable {
    Copyable copy();
}

interface Moveable {
    void move();
}

interface Shape extends Moveable {
    void click();
    double area();
}

class Point implements Copyable {
    private double x, y;
    public Point(double x, double y) { this.x = x; this.y = y; }
    public void move() { x += 1; y += 1; }
    public Point copy() { return new Point(x,y); }
}

class Circle implements Shape {
    private double x, y, r;
    public Circle(double r) { this.r = r; }
    public void click() { r += 1; }
    public void move() { x += 1; y += 1; }
    public double area() { return Math.PI * r * r; }
}
```

In jeder Aufgabe wird ein Objekt erzeugt, danach stehen mehrere mögliche Anweisungen. Welche der Anweisungen werden vom Java-Compiler ohne Fehlermeldung akzeptiert und liefern auch keine Fehler zur Laufzeit? Bitte wählen Sie alle gültigen Antwortmöglichkeiten aus.

Aufgabe 1.1.

3.75 / 5 Punkte

```
Shape circle = new Circle(0.3);
```

- [ ] `circle.move();`
- [x] `circle.area();`
- [ ] `Copyable c = circle.copy(); ((Circle)c).move();`
- [x] `circle.toString();`

Aufgabe 1.2.

5 / 5 Punkte

```
Point point = new Point(2.0,4.0);
```

- [ ] `point.click();`
- [x] `point.move();`
- [ ] `Shape s = ((Shape)point); s.move();`
- [x] `Copyable c = point.copy(); ((Point)c).move();`

Aufgabe 1.3.

5 / 5 Punkte

```
Copyable point = new Point(1.0,3.0);
```

- [x] `Copyable c = point.copy();`
- [x] `point.toString();`
- [ ] `point.move();`
- [x] `Point p = ((Point)point).copy();`

Aufgabe 1.4.

5 / 5 Punkte

```
Moveable circle = new Circle(12.0);
```

- [ ] `circle.click();`
- [x] `((Circle)circle).toString();`
- [x] `Object o = circle; ((Circle)o).area();`
- [x] `circle.move();`

Aufgabe 1.5.

5 / 5 Punkte

```
Circle circle = new Circle(3.0);
```

- [ ] `Moveable c = circle; c.click();`
- [x] `circle.move();`
- [x] `circle.area();`
- [x] `Object o = circle; ((Shape)o).area();`

2. Multiple-Choice-Aufgaben zu `equals` und `hashCode`

11.25 / 15 Punkte

In den folgenden Klassen sind die Implementierungen von `equals` und `hashCode` unvollständig. Ersetzen Sie die Buchstaben **A** und **B** durch einen oder mehrere der vorgeschlagenen Programmteile. Die Methoden müssen sich hinsichtlich der allgemeinen Bedingungen für `equals` und `hashCode` korrekt (und auch korrekt zueinander) verhalten. Bitte wählen Sie alle gültigen Antwortmöglichkeiten aus.

Aufgabe 2.1.

3.75 / 5 Punkte

```
class Box {
    final private Color color;
    final private int width;
    final private int height;
    final private int depth;

    // ...

    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj.getClass() != getClass()) return false;
        Box b = (Box)obj;
        A
    }

    public int hashCode() {
        B;
    }
}
```

A:

```
return b.color != null
    && b.color.equals(color);
```

B:

```
if (color == null) {
    return 0;
}
int hash = color.hashCode();
hash = hash * 31 + width;
hash = hash * 31 + height;
hash = hash * 31 + depth;
return hash;
```

A:

```
return b.color != null
    && b.color.equals(color)
    && (b.width * b.height * b.depth) == (width * height * depth);
```

B:

```
if (color == null) {
    return 0;
}
return color.hashCode() + (width * height * depth);
```

A:

```
return b.color != null
    && b.color.equals(color);
```

B:

```
if (color == null) {
    return 0;
} else {
    return 1;
}
```

☑ **A:**

```
return b.color != null
    && b.color.equals(color)
    && b.width == width
    && b.height == height
    && b.depth == depth;
```

B:

```
if (color == null) {
    return 0;
}
int hash = color.hashCode();
hash = hash * 31 + width;
hash = hash * 31 + height;
hash = hash * 31 + depth;
return hash;
```

Aufgabe 2.2.

3.75 / 5 Punkte

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

    // ...

    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj.getClass() != getClass()) return false;
        Vector v = (Vector)obj;
        A
    }

    public int hashCode() {
        B;
    }
}
```

☐ **A:**

```
return v.x == x;
```

B:

```
return y;
```

A:

```
return v.x == x && v.y == y;
```

B:

```
return x;
```

A:

```
return v.x == x;
```

B:

```
return x * y;
```

A:

```
return v.x == x && v.y == y;
```

B:

```
return x * y;
```

Aufgabe 2.3.

3.75 / 5 Punkte

```
class Car {
    final private String licensePlate;
    final private String model;

    public int currentSpeed() {
        return (int)(Math.random() * 200);
    }

    // ...

    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj.getClass() != getClass()) return false;
        Car c = (Car)obj;
        A
    }

    public int hashCode() {
        B
    }
}
```

- [ ] A:

```
return c.currentSpeed() == currentSpeed();
```

B:

```
int hash = currentSpeed();
hash = hash * 31;
return hash;
```

- [x] A:

```
return c.model != null && c.model.equals(model);
```

B:

```
if (model == null) {
    return 0;
}
return model.hashCode();
```

- [ ] A:

```
return c.model != null && c.model.equals(model);
```

B:

```
return 0;
```

- [ ] A:

```
return c.currentSpeed() == currentSpeed();
```

B:

```
return 0;
```