Test 3 in Programmkonstruktion

32.5 / 40 Punkte

Alle Aufgaben beziehen sich auf Java.

1. Multiple-Choice-Aufgaben zu Interfaces

25 / 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 Copyable {
    double area();
    void tap();
}

class Point implements Copyable, Moveable {
    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); }
    public void tap() { }
}

class Square implements Shape {
    private double x, y, w;
    public Square(double w) { this.w = w; }
    public void tap() { w += 1; }
    public void move() { x += 1; y += 1; }
    public double area() { return w * w; }
    public Square copy() { Square s = new Square(w); s.x = x; s.y = y; return s
}
```

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.

5 / 5 Punkte

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

- [ ] `((Shape)point).area();`
- [x] `Copyable c = point.copy(); c.toString();`
- [x] `point.tap();`
- [x] `point.move();`

Aufgabe 1.2.

5 / 5 Punkte

```
Square square = new Square(3.0);
```

- [x] `square.toString();`
- [x] `square.tap();`
- [ ] `((Moveable)square).area();`
- [ ] `((Moveable)square).move();`

Aufgabe 1.3.

5 / 5 Punkte

```
Moveable point = new Point(3.2,1.0);
```

- [ ] `Object o = point.copy(); o.toString();`
- [x] `point.move();`
- [x] `point.toString();`
- [ ] `point.tap();`

Aufgabe 1.4.

5 / 5 Punkte

```
Shape square = new Square(6.6);
```

- [ ] `((Moveable)square).move();`
- [x] `square.copy();`
- [x] `square.tap();`
- [x] `square.area();`

Aufgabe 1.5.

5 / 5 Punkte

```
Copyable square = new Square(2.4);
```

- [x] `((Shape)square).tap();`
- [x] `square.copy();`
- [x] `Square c = (Square)square.copy(); c.tap();`
- [x] `((Square)square).area();`

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

7.5 / 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.

2.5 / 5 Punkte

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

    // ...

    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 && v.y == y && v.z == z;
```

B:

```
return x + y;
```

- [ ] **A:**

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

B:

```
return x + y;
```

- [x] **A:**

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

B:

```
return x + y + z;
```

A:

```
return super.equals(obj);
```

B:

```
return x + y + z;
```

Aufgabe 2.2.

2.5 / 5 Punkte

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

    // ...

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

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

A:

```
return f.name != null
    && f.name.equals(name);
```

B:

```
return (int)size;
```

A:

```
return f.name != null
    && f.name.equals(name)
    && f.size == size;
```

B:

```
int hash = (int)size;
hash = (int)(hash * Math.random()) + name.hashCode();
return hash;
```

☑ **A:**

```
return f.size == size;
```

B:

```
int hash = (int)size;
hash = hash * 31 + name.hashCode();
return hash;
```

☐ **A:**

```
return f.name != null
    && f.name.equals(name)
    && f.size == size;
```

B:

```
return (int)size;
```

Aufgabe 2.3.

2.5 / 5 Punkte

```
class Watch {
    final private String serialNumber;
    final private boolean digital;

    public int currentTime() {
        return 49500;
    }

    // ...

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

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

- [x] A:

```
return w.serialNumber != null
    && w.serialNumber.equals(serialNumber);
```

B:

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

- [ ] A:

```
return w.serialNumber != null
    && w.serialNumber.equals(serialNumber)
    && w.digital == digital
    && w.currentTime() == currentTime();
```

B:

```
if (serialNumber == null) {
    return 0;
}
int hash = serialNumber.hashCode();
hash = (int)(hash * Math.random()) + (digital ? 1 : 0);
hash = (int)(hash * Math.random()) * currentTime();
return hash;
```

- [ ] A:

```
return w.digital == digital;
```

B:

```
return digital ? 1 : 0;
```

- [ ] A:

```
return w.serialNumber != null
    && w.serialNumber.equals(serialNumber);
```

B:

```
return currentTime();
```