Lösung

Aufgabe 1 (47.25 Punkte)

Item (1.5 Punkte)

```
// 1.5
public interface Item { // 1.0
    public double getVolume(); // 0.5
}
```

Freshness (3.5 Punkte)

```
// 3.5
public enum Freshness { // 0.25
    Fresh, // 0.25
    Okay, // 0.25
    Old; // 0.25

    public boolean canBeEaten() { // 0.5
        return this != Freshness.Old; // 2.0
    }
}
```

Vegetable (3.5 Punkte)

```
// 3.5
public class Vegetable implements Item { // 0.5
    public final String name; // 0.25
    public final double weight; // 0.25
    public final Freshness freshness; // 0.25

    public Vegetable(String name, double weight, Freshness freshness) { // 0.5
        this.name = name; // 0.25
        this.weight = weight; // 0.25
        this.freshness = freshness; // 0.25
    }

    public double getVolume() {  // 0.5
        return weight / 2; // 0.5
    }
}
```

FridgeCompartment (10.0 Punkte)

```
// 10.0
public class FridgeCompartment { // 0.5
```

```
    public final int length; // 0.25
    public final int width; // 0.25
    public final int height; // 0.25
    public final int level; // 0.25
    private List<Item> items; // 0.25

    public FridgeCompartment(int length, int width, int height, int level) { // 0.5
        this.length = length; // 0.25
        this.width = width; // 0.25
        this.height = height; // 0.25
        this.level = level; // 0.25
        this.items = new ArrayList<>(); // 0.25
    }

    protected double getVolume() { // 0.5
        return length * width * height; // 0.5
    }

    protected double usedSpace() { // 0.5
        double used = 0; // 0.25
        for (Item item : items) { // 0.5
            used += item.getVolume(); // 0.5
        }
        return used; // 0.25
    }

    private boolean hasEnoughSpace(Item item) { // 0.5
        return item.getVolume() < getVolume() - usedSpace(); // 0.5
    }

    protected void put(Item item) throws Exception { // 1.0
        if (hasEnoughSpace(item)) { // 0.5
            this.items.add(item); // 0.5
        } else {
            throw new Exception("Already Full"); // 0.5
        }
    }
}
```

IceCompartment (5.75 Punkte)

```
// 5.75
public class IceCompartment extends FridgeCompartment { // 0.5
    private final static int MINIMUM_STORAGE_TEMPERATURE = -10; // 0.25
    private int currentTemperature; // 0.25

    public IceCompartment(int length, int width, int height, int level, int
currentTemperature) { // 0.5
        super(length - 10, width - 10, height - 10, level); // 1.0
        this.currentTemperature = currentTemperature; // 0.25
```

```
    }

    public void put(Item item) throws Exception { // 0.5
        if (currentTemperature < IceCompartment.MINIMUM_STORAGE_TEMPERATURE && item
instanceof IceCream) { // 1.5
            super.put(item); // 0.5
        } else {
            throw new Exception("Not cold enough!"); // 0.5
        }
    }
}
```

VegetableCompartment (6.0 Punkte)

```
// 6.0
public class VegetableCompartment extends FridgeCompartment { // 0.5

    public VegetableCompartment(int length, int width, int height) { // 0.5
        super(length, width, height, 1); // 1.0
    }

    public void put(Item item) throws Exception { // 0.5
        if (item instanceof Vegetable vegetable && vegetable.freshness.canBeEaten()) {
// 1.5
            super.put(item); // 1.5
        } else {
            throw new Exception("Not eatable!"); // 0.5
        }
    }
}
```

Fridge (12.0 Punkte)

```
// 12.0
public class Fridge { // 0.5
    public final List<FridgeCompartment> compartments; // 0.25

    public Fridge(int numberOfCompartments, int length, int width, int height) { //
0.5
        ArrayList<FridgeCompartment> compartments = new ArrayList<>(); // 0.25
        compartments.add(new IceCompartment(length, width, height,
numberOfCompartments + 2, -5)); // 1.5
        compartments.add(new VegetableCompartment(length, width, height)); // 1.0
        for (int i = 0; i < numberOfCompartments; i++) { // 0.5
            compartments.add(new FridgeCompartment(length, width, height, 2 + i)); //
1.5
        }
        this.compartments = compartments; // 0.25
    }
```

```
    public boolean put(Item item) { // 0.5
        for (FridgeCompartment compartment : compartments) { // 0.5
            try { // 0.25
                compartment.put(item); // 0.5
                System.out.println("Placed on level " + compartment.level); // 0.5
                return true; // 0.25
            } catch (Exception exception) { // 0.25
                if (compartment instanceof IceCompartment || compartment instanceof
VegetableCompartment) { // 2.0
                    return false; // 0.25
                } else {
                    System.out.println(exception.getMessage()); // 0.5
                }
            }
        }
        return false; // 0.25
    }
}
```

ExamTask (5 Punkte)

```
// 5.0
public class ExamTask { // 0.5
    public static void main(String[] args) { // 0.5
        Fridge fridge = new Fridge(4, 90, 60, 90); // 0.5
        List<Item> items = new ArrayList<>(); // 0.5
        items.add(new IceCream("Magnum Mandel")); // 0.5
        items.add(new Vegetable("Tomaten", 500, Freshness.Fresh)); // 0.5
        items.add(new Vegetable("Rosenkohl", 250, Freshness.Old)); // 0.5
        for (Item item : items) { // 0.5
            if (!fridge.put(item)) { // 0.5
                System.out.println("Kein Platz gefunden."); // 0.5
            }
        }
    }
}
```