Lösung

Aufgabe 1 (65,25 Punkte)

Profile (7.5 Punkte)

```
// 7.5
public class Profile { // 0.5
  public Optional<String> phoneNumber; // 0.25
  public Optional<String> email; // 0.25

  public Profile(String phoneNumber) { // 0.5
    this(phoneNumber, null); // 0.5
  }

  public Profile(String phoneNumber, String email) { // 0.5
    this.phoneNumber = Optional.ofNullable(phoneNumber); // 0.25
    this.email = Optional.ofNullable(email); // 0.25
  }

  public String toString() { // 0.25
    String result = ""; // 0.5
    if (phoneNumber.isPresent()) { // 0.5
      result += phoneNumber.get(); // 0.25
    }
    if (email.isPresent()) { // 0.5
      if (result.length() != 0) { // 0.5
        result += " "; // 0.25
      }
      result += email.get(); // 0.25
    }
    return result; // 0.5
  }

  public boolean isValidated() { // 0.5
    return phoneNumber.isPresent() || email.isPresent(); // 0.5
  }
}
```

Person (3 Punkte)

```
// 3.0
public record Person(String name, int age, Map<Integer, Integer> visits, Profile
profile) {}
```

Area (3.0 Punkte)

```
// 3.0
public record Area(String name, int openingYear, double length, double width) {}
```

Flower (14.5 Punkte)

```
// 14.5
public record Flower(String name, Color color, int deliveryYear, Optional<String>
careInstructions) { // 3.0

    public static Predicate<Flower> hasAwesomeColor = // 1.0
            flower -> Color.getAwesomeColors().stream().anyMatch(color -> flower.
color().equals(color)); // 1.0

    public static Function<Flower, Flower> makeDangerous = // 1.0
            flower -> new Flower(flower.name(), Color.RED, 2025, flower
.careInstructions()); // 1.0

    public static Consumer<Flower> printHashvalue = // 1.0
            flower -> System.out.println("Hash: " + flower.hashCode()); // 1.0

    public static Predicate<Flower> needsNoSpecialCare = // 1.0
            flower -> flower.careInstructions().isEmpty(); // 1.0

    public static Predicate<Flower> deliverableBetween(int from, int to) { // 1.0
        return flower -> flower.deliveryYear() >= from && flower.deliveryYear() <= to;
// 1.0
    }

    public static Stream<Flower> getFlowers() { // 0.5
        return Stream.of( // 0.5
                new Flower("Daisy", Color.GREEN, 2023, Optional.of("sunlight"))); //
0.5
    }
}
```

Garden (27,75 Punkte)

```
// 27,75
public record Garden(List<Person> visitors, List<Area> areas, List<Flower> flowers) {
// 2.5

    // 3.75
    public List<String> q1() { // 0.25
        return visitors().stream() // 0.25
                .sorted((v1, v2) -> Integer.compare(v2.age(), v1.age())) // 1
                .limit(10) // 0.5
                .sorted((v1, v2) -> Integer.compare(v1.age(), v2.age())) // 1
                .map(visitor -> visitor.name().toUpperCase()) // 0.5
```

```
                .toList(); // 0.25
    }

    // 2.75
    public List<Person> q2() { // 0.25
        return visitors().stream() // 0.25
                .filter(
                        visitor -> visitor.visits().values().stream() // 1
                                .filter(value -> value > 3) // 0.5
                                .count() >= 2) // 0.5
                .toList(); // 0.25
    }

    // 2.75
    public Map<Integer, List<Person>> q3(int visitedYear) { // 0.5
        return visitors().stream() // 0.25
                .filter(
                        visitor -> visitor.visits().keySet().stream() // 1
                                .anyMatch(year -> year == visitedYear)) // 0.5
                .collect(Collectors.groupingBy(visitor -> visitor.age())); // 0.5
    }

    // 2
    public double q4() { // 0.25
        return areas().stream() // 0.25
                .filter(area -> area.openingYear() < 2015) // 0.5
                .mapToDouble(area -> area.width() * area.length()) // 0.5
                .sum(); // 0.5
    }

    // 1.75
    public List<Integer> q5() { // 0.25
        return visitors().stream() // 0.25
                .map(Person::age) // 0.5
                .distinct() // 0.5
                .toList(); // 0.25
    }

    // 4.0
    public long q6() { // 0.25
        return visitors().stream() // 0.25
                .map(
                        visitor -> visitor
                                .visits() // 0.5
                                .values()
                                .stream() // 1
                                .mapToInt(Integer::intValue) // 0.5
                                .sum()) // 0.5
                .mapToInt(Integer::intValue) // 0.5
                .sum(); // 0.5
    }
```

```
    // 2
    public void q7() { // 0.25
        flowers().stream() // 0.25
                .filter(Flower.hasAwesomeColor) // 0.5
                .map(Flower.makeDangerous) // 0.5
                .forEach(Flower.printHashvalue); // 0.5
    }

    // 2
    public List<Flower> q8(int from, int to) { // 0.5
        return flowers().stream() // 0.25
                .filter(Flower.deliverableBetween(from, to)) // 0.5
                .filter(Flower.needsNoSpecialCare) // 0.5
                .toList(); // 0.25
    }

    // 2.5
    public Map<String, String> q9() { // 0.25
        return flowers().stream() // 0.25
                .collect(Collectors.toMap( // 0.5
                        flower -> flower.name(), // 0.5
                        flower -> flower.careInstructions().orElse("")) // 1.0
                );
    }

    // 1.75
    public List<Person> q10() { // 0.25
        return visitors().stream() // 0.25
                .filter(visitor -> visitor.profile().isValidated()) // 1.0
                .toList(); // 0.25
    }
}
```

ExamTask (9.5 Punkte)

```
// 9.5
public class ExamTask { // 0.5
  public static void main(String[] args) { // 0.5
    Map<Integer, Integer> aliceVisits = new HashMap<>(); // 0.5
    aliceVisits.put(2023, 5); // 0.25
    aliceVisits.put(2024, 2); // 0.25

    Person alice = new Person("Alice", 45, aliceVisits, new Profile("11880",
"a@b.de")); // 1
    Person bob = new Person("Bob", 30, new HashMap<>(), new Profile("0815")); // 1.5
    List<Person> visitors = List.of(alice, bob); // 0.5

    List<Area> areas = List.of( // 0.5
        new Area("Rose Garden", 2010, 10.0, 15.0), // 0.5
```

```
        new Area("Tropical House", 2018, 8.0, 12.0)); // 0.5

    List<Flower> flowers = Flower.getFlowers().toList(); // 0.5

    Garden garden = new Garden(visitors, areas, flowers); // 0.5

    garden.q9() // 0.5
        .entrySet().stream() // 0.5
        .forEach( // 0.5
            System.out::println); // 0.5
  }
}
```