Questions and Exercises

These questions and exercises is an opportunity to see what you've learnt from the lecture as well as practice the new things we've been talking about. In other words, these questions and exercises are completely optional but it's recomended to do them. In the end of the document you will find the answers to the questions as well as possible solutions to the exercises, note that one can solve an exercise in different ways. There will also be some suggestions about what one could code if one want to continue with some more advanced things. These suggestions will not come with a possible solution and might include things that haven't been covered in the lecture.

Question 1
What is a constructor? And what is the syntax difference between a constructor and a method?

Question 2
Fields are a type of variables. What's so special about them?

Question 3
The static keyword can be used in front of methods and fields. What is it used for?

Exercise 1
Write a person class. A person should have a name, an occupation and an age. Write a simple program which uses this class to group a bunch of persons by their occupation. Inside each group the persons should be sorted by their age. For example, given the following persons

Dan, Janitor age 32
Paul, Postman age 25
Caroline, Postman age 23
Bill, Janitor age 38

the code should output it as

Janitor
Bill, age 38
Dan, age 32

Postman
Paul, age 25
Caroline, age 23

Exercise 2
Write an occupation class. An occupation should have a company, a position in the company, a salary and the amount of hours of work per month. Also write a method for the hourly salaries of

the occupation.

Write a program that prints out the average hourly salaries for each comapny. You can assume that the list is already grouped by company. For example, given the following occupations

Steve's Carts, Manager, 200000, 1
Steve's Carts, Mechanic, 1200, 300
Steve's Arcades, Manager, 25000, 40
Steve's Arcades, Janitor, 1000, 160

the code should print out the following

Average of hourly salaries at Steve's Carts is 100002
Average of hourly salaries at Steve's Arcades is 315

Further explorations
Use the Chair class from the lecture, the Person class from exercise 1 and the Occupation class from exercise 2. Change it so the Chair class is refering to the Person class and so the Person class is reffering to the Occupation class.

Write a program that allows persons to sit down around a table for a board meeting. When the meeting is about to start make sure that no body from another company is present. Also make sure that there's no body there who's not a board member and finally make sure that there's no kid around.

Expand the program by creating a Meeting class. This class should include a list of all the chairs as well as the restrictions of the meeting, for instance age limits, company restrictions and company position restrictions. Use this meeting class to allow the program to handle different kinds of meetings and to be able to handle multiple meetings at the same time. Make sure that the same person is not attending multiple meetings at one given time.

Answers and solutions

Answer to Question 1
A constructor is a special type of method. It is called when a new object is being created. This means that the parameter lists of the constructors defines how you can create the object. The difference, syntax-wise, to a method is that the constructor has no return type and must be named the same as the class.

Answer to Question 2
Fields are variables directly in the class, rather than in a method. The only special thing about them is where they are declared. This makes them accessible from within all the methods of the class.

Answer to Question 3
The static keyword makes a method or a field indepentent of objects. This means that you can call it directly from the class, opposed to have to define which object to refer to when running the method or accessing the field. The static keyword is left out more often than it's used.

Possible solution to Exercise 1

```
//the class
public class Person {

   //fields
   String myName;
   String currentOccupation;
   int currentAge;

   //constructor
   Person(String name, String occupation, int age) {
      myName = name;
      currentOccupation = occupation;
      currentAge = age;
   }

   //methods
   String getName() {
      return myName;
   }

   String getOccupation() {
      return currentOccupation;
   }

   int getAge() {
     return currentAge;
   }
}

//the program using the class
public class PersonExample {

   public static void main(String[] args) {
      Person[] persons = {
         new Person("Dan", "Janitor", 32),
         new Person("Paul", "Postman", 25),
         new Person("Caroline", "Postman", 23),
         new Person("Bill", "Janitor", 38)
      };

      //sort the list
```

```
        sort(persons);

        //print it out
        print(persons);
    }

    static void sort(Person[] persons) {
        for (int i = 0; i < persons.length; i++) {
            for (int j = persons.length - 1; j > i; j--) {
                Person person1 = persons[j];
                Person person2 = persons[j - 1];

                //alphabetically compare the occupations
                int comparison = person1.getOccupation().compareTo(person2.getOccupation());

                if (comparison < 0 || (comparison == 0 && person1.getAge() > person2.getAge())) {

                    //if the values are in the wrong order, swap them
                    Person temp = persons[j];
                    persons[j] = persons[j - 1];
                    persons[j - 1] = temp;
                }
            }
        }
    }

    //print out a sorted list of persons, grouped by their occupation
    static void print(Person[] persons) {
        String lastOccupation = "";
        for (int i = 0; i < persons.length; i++) {
            if (lastOccupation.equals("") || !lastOccupation.equals(persons[i].getOccupation())) {
                System.out.println("\n" + persons[i].getOccupation());
            }
            System.out.println(persons[i].getName() + ", age " + persons[i].getAge());

            lastOccupation = persons[i].getOccupation();
        }
    }

}
```

Possible solution to Exercise2

```
//the class
public class Occupation {

    //fields
    String myCompany;
    String myPosition;
```

```
    int currentSalary;
    int hoursPerMonth;

    //constructor
    Occupation(String company, String position, int salary, int hours) {
        myCompany = company;
        myPosition = position;
        currentSalary = salary;
        hoursPerMonth = hours;
    }

    //methods
    String getCompany() {
        return myCompany;
    }

    String getPosition() {
        return myPosition;
    }

    int getSalary() {
        return currentSalary;
    }

    int getHours() {
        return hoursPerMonth;
    }

    int getHourlySalary() {
        return getSalary() / getHours();
    }
}

//the program using the class
public class OccupationExample {
    public static void main(String[] args) {
        Occupation[] occupations = {
            new Occupation("Steve's Carts", "Manager", 200000, 1),
            new Occupation("Steve's Carts", "Mechanic", 1200, 300),
            new Occupation("Steve's Arcades", "Manager", 25000, 40),
            new Occupation("Steve's Arcades", "Janitor", 1000, 160)
        };

        int totalHourlySalary = 0;
        int occupationCount = 0;
        for (int i = 0; i < occupations.length; i++) {
            totalHourlySalary += occupations[i].getHourlySalary();
            occupationCount++;
```

```
            if (i == occupations.length - 1 || !occupations[i].getCompany().equals(occupations[i +
1].getCompany())) {
            System.out.println("Average of hourly salaries at " + occupations[i].getCompany() + " is
" + (totalHourlySalary / occupationCount));
            totalHourlySalary = 0;
            occupationCount = 0;
        }
    }
  }
}
```