Assignment 2

Topics: Classes, Initialization Blocks, Equality and Cloning of Objects

NOTE: exercises with prefix * are not mandatory**

1. Please read the tutorials at this link:
   http://docs.oracle.com/javase/tutorial/getStarted/intro/index.html
   http://docs.oracle.com/javase/tutorial/java/concepts/index.html
   **** http://java.sun.com/docs/books/jvms/ (for highly interested readers)*

2. Create a class *Student* that has data fields *name*, *group*, *proficiencyInJava*, *cprNumber* and *gender*. Implement a few constructors for the class, the getter and setter methods, and the *toString()* method.

3. Create a class *TestStudent* to test the class *Student*. *TestStudent* will therefore contain the *public static void main()* method. In *main()* create 5 *Student* objects and play around with them (e.g. set different values for the data fields of the single objects, print out each of them, etc.)

4. Add a method *boolean hasSameFluencyInJavaAs(Student anotherStudent)* that checks whether two students have the same proficiency in Java and test it in the *TestStudent* class

5. Implement a method that tests whether two students are the same student. To do so you have to override the *equals()* method in class *Student*. Test your method *equals()* in the *main()* in *TestStudent*. Comment out your method *equals()* in class Student and run again *TestStudent*. Is the result of your *equals()* different than the default *equals()*? Please explain.

6. Implement a search method that prints out a list of *Student* objects for which the *proficiencyInJava* field in class *Student* is set to a certain value. This should look something like: *void studentsWithProficiency(int proficiency)* (here, I assume you chose *int* as the data type of the data member *proficiencyInJava*), but your method can also have a different signature or return value. Think through where (i.e. in which class) to place such a method and why.

7. ***As in 6 but instead of printing out objects store them into an array

8. In *Student* class, add data members to store all the grades for all the courses taken by a student. Also implement a method *float averageNote()* that returns the average grade over all courses taken by a student

9. Override the *clone()* method in *Student*. Test your method in *TestStudent* by creating an object of type *Student*, printing it out, cloning it, and printing out that clone. If you implemented *clone()* correctly, the two print outs should print the same values.

10. Imagine in class *Student* you have the following:

```
public class Student {
        String name;
        public Student(String name) {
                this.name = name;
        }
        …
}
```

Change *this.name = name* into *name = this.name* and comment on the result. Now change *name = this.name* into *name = name* and comment on again the new result.

11. Remove the keyword *static* in method *public static void main(String[] args)* in class *TestStudent*. Now, try to run *TestStudent* and comment on the result.

12. ***Add a static field in class *Student* to keep track of the number of *Student* objects created. Anytime, a new *Student* object is created, you should print out a message like "New Student added! There are a total of 6 students" when the sixth *Student* object is created.

13. Create a JavaDoc documentation for your code.

14. ***To write a method that accepts an arbitrary number of arguments of different type when it is called, one must specify the last parameter as follows: *Objects … args*

    Try out this piece of code (or some similar one)

```
public class freeArgs {

   public static void main(String[] args) {

        Student student = new Student("Elisa");
        printAllThese(1, "one", 7, "seven", "I am done"); // 5 arguments
        printAllThese ("Anything", 3E3, true, student); // 4 arguments
   }

   public static void printAllThese(Object ... args) {
      for (Object arg : args) {
        System.out.print(" " + arg);
      }
      System.out.println();
   }

}
```

    Why does the *Object … args* text have to be the last parameter? What happens if it is not the last one?

15. Given three classes *PhDStudent*, *TAStudent* and *Student*, write a program that takes as command line argument a *String* like "TA1 TA3 ST3 TA4 PHD6". This command line has to be interpreted as: TA1 = teaching assistant 1, ST3 = regular student 3, PHD6 = PhD student 6, where the number indicates the position within an array of students, teaching assistants and PhD students. Your application must print out these objects

(NOTE: you need to have some pre-stored objects before printing them out; since we cannot deal with files now, simply create some objects, store them into arrays, and then access them).

16. ***Look at these two classes:

```
public class InitBlockTest {
        private int[] myArray = new int[C_SIZE];
        static int C_SIZE = 100;
}

public class InitBlockTest {
        private int[] myArray = new int[C_SIZE];
        int C_SIZE = 100;
}
```

Without entering the examples above into any IDE, could you say if any of them is correct or not? And if so, which one is correct or incorrect, and why?
Type in the code above in NetBeans and see if it complains. If you get an error message, have a look at the error message as well.

17. ***Look at this class:

```
public class InitBlocks {

    private static int a = 5;
    private static int b;
    private int c;

    public  static void infoStatic() {
        System.out.println("a: " + a);
        System.out.println("b: " + b);
    }

    public void info() {
        System.out.println("a: " + a);
        System.out.println("b: " + b);
        System.out.println("c: " + c);
    }

    {
        System.out.println("Non static block initialized!");
        b = a * a * a;
        c = 1234567;
    }

    static {
        System.out.println("Static block initialized!");
        b = a * a;
    }
    public static void main(String args[]) {
        InitBlocks.infoStatic();
        InitBlocks ib = new InitBlocks();
        InitBlocks.infoStatic();
        ib.info();
    }
}
```

Without entering the examples above into any IDE, could you say if the code can run? If yes, what is the output? If no, why and how can you fix it?
After you have answered these questions, type in the code above in NetBeans and see what happens when you run it.