Assignment 1

Programming II / Enterprise Java Programming, Year 2019/2020, Semester 2

- *If asked to write Java code, submit source code files (filename ending .java). Please create a separate file for each of your Java classes or interfaces.*
- *Put all files into a single zip-file named "Assignment1-YourName.zip" and submit via Blackboard.*
- ***Deadline (strict): Wednesday, 5th February, 23:59. Late submission only with a medical certificate.***
- *Please note that the graders are advised to check all submissions for plagiarism. Assignments need to be solved individually (no group work).*
- *If you have questions about the assignment or need help with it, the best approach is to ask a tutor during one of the lab sessions (it is recommended to visit all lab sessions).*
- ***Use (Java) comments to explain your code. Missing or insufficient comments lead to mark deductions.***

Q1. The following (not entirely realistic…) code is correct, but partially redundant and difficult to maintain:

```
class Workaholic {

   public static final int RETIREMENT_AGE = 65;
   public static final int OVERTIME = 500;

   private String name = null;
   private int age = 0;
   private float earned = 0.0f;
   private float hourlyIncome = 0.0f;

   public Workaholic(String name, float hourlyIncome, int age) {
          this.name = name;
          this.hourlyIncome = hourlyIncome;
          this.age = age;
   }

   public void work(int hours) {

          for(int i = 1; i<=hours; i++)
                 earned += hourlyIncome;

          for(int j = 1; j<=OVERTIME; j++)
                 earned += hourlyIncome;

   }

   public void work() {
          while(age++ < RETIREMENT_AGE)
                 work(2000);
   }

   public String info() {
          return name + " earned " + earned;
   }
}

class Worker {

   public static final int RETIREMENT_AGE = 65;
   private String name = null;
   private int age = 0;
   private float earned = 0.0f;
   private float hourlyIncome = 0.0f;

   private Worker coWorker = null;
```

PTO

```
    public Worker(String name, float hourlyIncome, int age, Worker coWorker) {
           this.name = name;
           this.hourlyIncome = hourlyIncome;
           this.age = age;
           this.coWorker = coWorker;
    }

    public void work(int hours) {

           for(int i = 1; i<=hours; i++) {

                  earned += hourlyIncome;

                  if(coWorker!=null && i%5==0)
                         delegate(1); // from time to time

           }
    }

    public void work() {

           while(age++ < RETIREMENT_AGE)
                  work(1600);

    }

    private void delegate(int hours) {

           coWorker.work(hours);

    }

    public String info() {

           return name + " earned " + earned;

    }

    public static void main(String[] args) {

           Worker jane = new Worker("Jane", 20, 25, null);
           Worker john = new Worker("John", 20, 45, jane);
           Workaholic bill = new Workaholic("Bill", 20, 25);

           john.work();
           jane.work();
           bill.work();

           System.out.println(john.info());
           System.out.println(jane.info());
           System.out.println(bill.info());

    }
}
```

Simplify the given code as much as possible using class inheritance (but do not remove classes or functionality of the given classes). Also, simplify the loops in the given code, where possible. Make sure that your new versions of the classes yield the same results as the old versions when used in `main()`. **[30 marks]**

PTO

(For solving the following questions you will need knowledge from CT545 lectures, and also from Semester 1.)

Q2. Create a public interface `ConsoleIO` . The interface should declare two methods: `updateFromConsole` and `writeToConsole`. These methods should have neither parameters nor return values.

Also, create a *generic* interface `ComparableGeometricFigure<…>` which extends built-in interface `Comparable<…>` and which ensures, using generics, that only objects of subclasses of `GeometricFigure2` are comparable with each other (but it should not be restricted to comparing octagons, but should also be usable with other subclasses of `GeometricFigure2`). `GeometricFigure2` is given in the lecture notes (lecture 1b).

Create a class `Octagon` for regular octagons (symmetric octagons where all sides have the same length) which extends `GeometricFigure2` and implements your interfaces `ConsoleIO` and `ComparableGeometricFigure<…>`.

`writeToConsole` should print the field values and the area of the octagon. `updateFromConsole` should prompt the user interactively for a number (hint: `java.util.Scanner`) and update the side length of this octagon to this value.

Your implementation of `compareTo` in `Octagon` should compare two octagons in terms of their areas. Make sure that your `compareTo` implementation can compare only octagons with each other, and that the Java compiler ensures this at compile time (again, using generics).

Finally, add a static `main` method with some appropriate code which demonstrates the use of `updateFromConsole`, `compareTo` and `writeToConsole`. **[40 marks]**

Q3. Modify class `Circle` from the lecture so that it becomes a subclass of `GeometricFigure2` (which is also given in the lecture notes). Then, modify `Circle` so that it becomes a *generic* class with a type parameter T which represents any subclass of a new abstract class `Number` (see below). Within your new version of class `Circle`, T should be used everywhere in place of `double` (i.e., as the type of the circle's radius and area).

Create an appropriate abstract class `Number` and two non-abstract subclasses of `Number`: `NumberDouble` and `NumberInt`. Objects of `NumberDouble` should hold double values whereas objects of `NumberInt` should hold integer values. These classes should have constructors for creating such objects from primitive values of type `double` respectively `int`. Also, add a static `main`-method to `Number` which creates a circle object (with type argument `NumberDouble` and radius 6.9) and prints the result of calling `calcArea()` on that object.

Remarks: You might also need to modify a particular line of code in class `GeometricFigure2`.
Avoid type casts as far as possible (but one particular type cast might be unavoidable…!). **[30 marks]**