Question Sheet 1

Name: _____ ID: _____

These questions do not have a formal, definitive answer. They are meant to be food for thoughts. Feel free to seek answers on browsing the Internet, talking to other software developers or reading books.

1. Which factors compromise the maintainability of software?

2. How to improve the following program:

```
class Employee {
  private int hours;
  public Employee(int hours) {
    this.hours = hours;
  }
  public int getHours() {
    return hours;
  }
}
```

```
class Manager extends Employee {
  private int numSubordinates;
    public Manager(int hours, int numSubs) {
      super(hours);
      numSubordinates = numSubs;
    }
    public int getNumSubs() {
      return numSubordinates;
    }
  }
```

```
class Guard extends Employee {
  private double dangerFactor;
  public Guard(int hours, double df) {
    super(hours);
    dangerFactor = df;
  }
  public double getDangerFactor() {
    return dangerFactor;
  }
}
public class Payment {
  public double getPay(Employee e) {
    if (e instanceof Manager) {
```

```
      return 1000.0 * ((Manager)e).getNumSubs() + 200.0 * e.getHours();
    } else if (e instanceof Guard) {
      return 800.0 * ((Guard)e).getDangerFactor() + 160.0 * e.getHours();
    } else {
      return 600.0 * e.getHours();
    }
  }
  public static void main(String args[]) {
  }
}
```

3. The program below contains a well known logical error, called "The circle-ellipses dilemma". Describe this error, and explain how it could be fixed.

```
interface Point {
  int getX();
  int getY();
  int setX(int x);
  int setY(int y);
}
public class Ellipse {
  private Point f1, f2;
  public Point getF1() { return f1; }
  public Point getF2() { return f2; }
  public void setF1(Point f) { f1 = f; }
  public void setF2(Point f) { f2 = f; }
}
public class TestShape {
  public static void stretchX(Ellipse e, int w) {
    Point f1 = e.getF1();
    Point f2 = e.getF2();
    f1.setX(f1.getX() - w);
    f2.setX(f2.getX() + w);
  }
  public static void main(String args[]) {
    Ellipse e = new Circle();
    stretchX(e, 10);
  }
}
```

4. The dependence inversion principle says: "depend upon Abstractions. Do not depend upon implementations". What is the motivation behind this principle?

5. Code Java program that reads characters from the standard input, and then prints these characters in the output.

6. We now have an object called `ScreenOutput`, which has a method `printOnScreen(int)`. Modify the program above to handle this object too.

7. We now have an object called `FileInput`, which a method `readNextChar()`. Modify the program above to handle this object.

8. Ok, what is the best design for that program that reads characters from the input, and prints them into the output?

9. The board of directors of *Toy Inc.* is concerned about the safety of the company, and so they decided to buy a software system that controls the doors of the buildings. Each door has three methods: `lock`, `unlock` and `isOpen`. If the door remains open for a certain time, then an alarm will fire off. This alarm will notify a set of client objects. Each client must register itself in the door, to receive notifications. The door has, thus, a method `register(TimerClient)`.

   Use a class diagram to represent this system.

10. The board of directors of *Toy Inc.* was so pleased with your solution to the previous exercise (whatever it may be). In particular, they really liked the event notification system, and now they want to port this service to the main assemble line. The directors want each toy having a method `register(TimerClient)`, which is activated when the toy is ready to be taken away from the assemble line. How to reuse the code in the previous question in this new scenario?