CENG 305
Object Oriented Programming With Java
Fall 2013
HOMEWORK 3

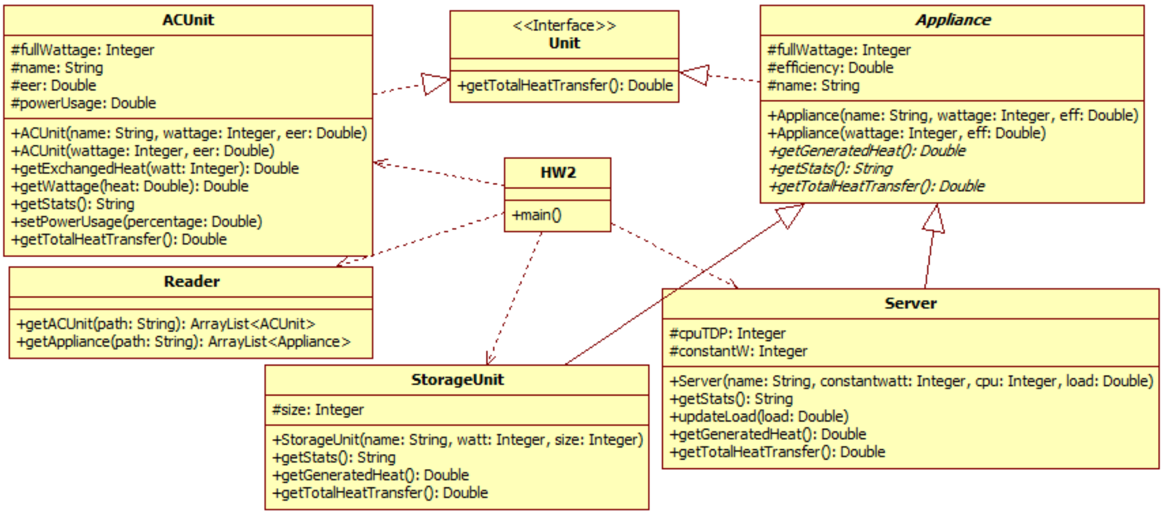
1. Regulations

- Due Date: 13 / 01 /2014
- Programming Language: You will use Java (version 1.6) to implement the assignment.
- Submission: Submit your homework through Cow (https://cow.ceng.metu.edu.tr).You will submit a ZIP file named "e1xxxxxx.zip", which will include "Appliance.java" file, "ACUnit.java" file, "Server.java" file, "StorageUnit.java" file, "Reader.java" file, "Unit.java" file.
- The homework should be done and submitted individually.
- Late Submission: You are allowed to submit your homework late with a penalty of 5*days*days.
- Cheating: The homework has to be done individually. In case of cheating, all involved (source(s) and receiver(s)) will get zero.
- Newsgroup: Check the metu.ceng.course.305 for discussions, announcements, etc. about the homework, regularly.
- Grading: Black box method will be used for evaluating and grading your assignments. Hence, please check sure that your codes can be compiled and run with the adequate output before submitting them. Furthermore, adequate indentation of the code, documentation and following submission rules are important since they will be considered in grading.

2. Specification

In this homework you will add an interface and make modification to the system you implemented for the previous homework.

You will turn the `Appliance` class into an abstract class, `getGeneratedHeat`, `GetStats`, `GetTotalHeatTransfer` functions are all abstract functions. You will also implement the Unit interface which enforces the `getTotalHeatTransfer` function.


ACUnit
#fullWattage: Integer
#name: String
#eer: Double
#powerUsage: Double
+ACUnit(name: String, wattage: Integer, eer: Double)
+ACUnit(wattage: Integer, eer: Double)
+getExchangedHeat(watt: Integer): Double
+getWattage(heat: Double): Double
+getStats(): String
+setPowerUsage(percentage: Double)
+getTotalHeatTransfer(): Double
<<Interface>>
Unit
+getTotalHeatTransfer(): Double
Appliance
#fullWattage: Integer
#efficiency: Double
#name: String
+Appliance(name: String, wattage: Integer, eff: Double)
+Appliance(wattage: Integer, eff: Double)
+getGeneratedHeat(): Double
+getStats(): String
+getTotalHeatTransfer(): Double
HW2
+main()
Reader
+getACUnit(path: String): ArrayList<ACUnit>
+getAppliance(path: String): ArrayList<Appliance>
StorageUnit
#size: Integer
+StorageUnit(name: String, watt: Integer, size: Integer)
+getStats(): String
+getGeneratedHeat(): Double
+getTotalHeatTransfer(): Double
Server
#cpuTDP: Integer
#constantW: Integer
+Server(name: String, constantwatt: Integer, cpu: Integer, load: Double)
+getStats(): String
+updateLoad(load: Double)
+getGeneratedHeat(): Double
+getTotalHeatTransfer(): Double

Appliance class

Appliance class implements the Unit interface.

Operation

You will turn `getGeneratedHeat`, `GetStats`, `GetTotalHeatTransfer` functions are all abstract functions, they should be implemented in the child classes. You can directly copy `getGeneratedHeat` function to child classes.

ACUnit Class

Attributes

`powerUsage: Double` is the percentage of the maximum power is used by the unit.

Operations

`setPowerUsage(percentage: Double)` Used to set the level the AC unit is working. For example if it is 0,50 unit is using half of its maximum power.

`getTotalHeatTransfer(): Double` Returns the total heat transferred by this unit. It is a negative value for AC units.

Server Class

Operation

`getTotalHeatTransfer(): Double` Returns the total heat transferred by this unit. It is a negative value for servers.

StorageUnit Class

Operation

`getTotalHeatTransfer(): Double` Returns the total heat transferred by this unit. It is a negative value for storage units.

Reader Class

You are expected to use the exception handling methods in this class. Class itself should not throw an exception, it should be handled within class. An empty List can be returned in case of an error that cannot be recovered e.g. "file does not exist" .

HW1 Class

This class will only contain the main function. You can use the following function. You should look at the usage of `System.out.format` function and the `instanceof` key word. Also you should look at how `Unit` class is used to keep both `Appliance` and `ACUnit` class in the `ArrayList`.

```
public static void main(String[] args) {
        try {
            ArrayList<Unit> la = new ArrayList<>();
            la.addAll(Reader.getAppliances("txt.txt"));
            la.addAll(Reader.getACUnit("txt.txt"));
            Double heat = 0.0;
            for (Unit u: la) {
                heat += u.getTotalHeatTransfer();
            }
            System.out.format("Total heat difference in the server
room after unit time:%.3f%n", heat);
            System.out.println("Changing AC unit levels...");
            for (Unit u: la){
                if (u instanceof ACUnit)
                {
                    ACUnit n = (ACUnit) u;
                    n.setPowerUsage(0.85);
                }
            }
            for (Unit u: la) {
                heat += u.getTotalHeatTransfer();
            }
            System.out.format("Total heat difference in the server
room after unit time:%.3f%n", heat);

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
```

3. Example File

Server
serverbir
500
50
0.30
StorageUnit
storageunitbir
500
500
ACUnit
klimabir

```
250
11.0
```

Sample output:

```
Total heat difference in the server room after unit time:-158,912
Changing AC unit levels...
Total heat difference in the server room after unit time:94,677
```

4. Submission

You will submit a single exxxxxxx.zip file. File name have to be your user name. Only your class files ("Appliance.java", "ACUnit.java" "Server.java", "StorageUnit.java", "Reader.java", "Unit.java")) and generated java documentation have to be in this .zip file. All the package, class, attribute and operation names have to be **exactly** the same with the names given in this document. If you do not follow the submission rules a penalty will be applied to your grade.