CSE 102L Computer Programming Laboratory – Exercise 4

Due Date: 23:59 Thursday April 6th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *Ex4_yourStudentNumber.java*. You are not required to provide a main method, however you can write one to test your code. Submissions made after the deadline will not be accepted, be sure to submit your work before the due date. Your code will be automatically controlled, so be sure to have the same class, method, variable names. Failure to do so may result in you receiving 0 from this exercise. All classes should be written to a single Java file. Do not zip your java file, Do not use package in your source code, Do not include @author and @since information.

Exercise

Write the following classes according to given specifications and **declare all attributes as private except cpu and ram, declare those as protected in Computer class:**

1. Computer
   - Attributes:
     - cpu: CPU
     - ram: RAM
   - Methods:
     - Constructor that takes *cpu* and *ram*
     - run(): None, this method gets all the values in the diagonal in ram (using the appropriate method from ram) and sends them to *cpu* for computation (*compute* method of cpu) and store the result of the computation in [0,0] memory location. Since *compute* method only sums given two integers, all this *run* method does is summing all the values in diagonal of ram and storing the result in [0,0] memory.
     - toString(): String, format should be: “Computer: ” + cpu + “ “ + ram

2. Laptop – child of Computer
   - Attributes:
     - milliAmp: int
     - battery: int
   - Methods:
     - Constructor that takes *cpu*, *ram*, and *milliAmp* and sets *battery* of laptop to %30 of its maximum capacity(*milliAmp*)
     - batteryPercentage(): int, returns the remaining battery percentage. For example it should return 30 when we instantiate a laptop object and use its batteryPercentage method.

- charge(): None, charges the *battery* until it reaches 90% with 2% increments
- run(): None, overrides the *run* method of its superclass to only *run* if its battery is more than 5% and reduce the battery by 3%, otherwise charges the battery.
- toString(): String, format should be: super.toString() + " " + battery

3. Desktop – child of Computer
   - Attributes:
     - peripherals: java.util.ArrayList<String>
   - Methods:
     - Constructor that takes *cpu*, *ram*, and varargs *peripherals* as String array. Do not use a loop to add each element to array list, use a method from java.util.Arrays class to cast the array to array list.
     - run(): None, overrides its super class's implementation to compute the sum (use compute method of cpu, don't use '+' operator) of all elements in the ram and stores the result in [0,0].
     - plugIn(String peripheral): None, adds the new peripheral to list
     - plugOut(): String, returns and removes the last peripheral in the list
     - plugOut(int index): String, returns and removes the peripheral at the index location of the list.
     - toString(): String, super.toString() + each peripheral space separated

4. CPU
   - Attributes:
     - name: String
     - clock: double
   - Mehods:
     - Constructor that takes *name* and *clock*
     - Accessor methods of *name* and *clock* (not mutator)
     - compute(a: int, b: int): int, sum given *a* and *b* and return it
     - toString(): String, in the format "CPU: " + name + " " + clock + "Ghz"

5. RAM
   - Attributes:
     - type: String
     - capacity: int
     - memory: int[][]
   - Methods:
     - Constructor that takes *type* and *capacity*. Uses *initMemory()* method to initialize the *memory*.

- Accessor methods for *type* and *capacity* (not mutator)
- initMemory(): None, this method should be **private**. Creates a 2D array with *capacity*. So if *capacity* is 16 you will have 16x16 matrix. Fills the memory with random numbers ranging from 0 to 10 (use Random class).
- check(i: int, j: int): boolean, this method should be **private**. Checks if given location with given *i* and *j* is out of bounds for *memory* or not. Returns false if it is out of bounds, true otherwise.
- getValue(i: int, j: int): int, gets the value from *memory*. This method should use *check* to see if it is out of bounds or not. If it is returns -1, otherwise returns the value.
- setValue(i: int, j: int, value: int): None, stores the given value at given location in *memory*. This method should use *check* to see if it is out of bounds or not.
- toString(): String, in the format of: “RAM: “ + *type* + “ “ + *capacity* + “GB”