

CSE 102L Computer Programming Laboratory – Homework04

Due Date: 23:59 Tuesday April 7th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *HW04_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 homework. 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.
ComputationException and MemoryException is throwable from this method. Catch the **ComputationException** from *cpu* instance and handles with **fixComputation** method using exception instance. Catch **InterruptedException** and print the stack trace.
 - **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`
- Methods:
 - 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.
 - Wait with **`Thread.sleep((long)(5 / clock * 1000))`**
 - **`ComputationException`** and **`InterruptedException`** is throwable from this method. Catch **`InterruptedException`** and rethrow it message: "Exception occurred due to internal clock speed: " + `clock`
 - If computation result is less than 0, throw **`ComputationException`** with **`this, a, b`**.
 - `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**. **MemoryException** is throwable. Checks if given location with given *i* and *j* is out of bounds for *memory* or not. Returns true if it is **not** out of bounds, **throw MemoryException** with **this**, *i*, and *j* otherwise.
 - *getValue(i: int, j: int)*: int, gets the value from *memory*. **MemoryException** is throwable. This method should use *check* to see if it is out of bounds or not.
 - *setValue(i: int, j: int, value: int)*: None, stores the given value at given location in *memory*. **MemoryException** is throwable.
 - *toString()*: String, in the format of: "RAM: " + *type* + " " + *capacity* + "GB"

6. MemoryException, unchecked exception

- Attributes:
 - ram: RAM
 - address1: int
 - address2: int
- Methods:
 - Constructor that takes ram, address1, and address2, sets data fields accordingly. Uses 1 parameter constructor of super's with message "Memory out of range! With addresses #[address1, address2]"

7. ComputationException, checked exception

- Attributes:
 - cpu: CPU
 - value1: int
 - value2: int
- Methods:

- Constructor that takes cpu, value1, and value2. Uses 1 parameter constructor of super's with message "Computation exception occurred on " + cpu + " with values: (value1, value2)"
- Constructor that takes a single ComputationException exception instance **e** as parameter and uses 1 parameter constructor of super's with message "Unhandled exception occurred at " + e.cpu + " with values " + e.value1 + " and " + e.value2 + ":\n\t" + e.getMessage()
- fixComputation(val1: int, val2: int): int, takes the absolute of both values and uses cpu instance's compute method. Catches **ComputationException** and rethrows. Catches **InterruptedException** and prints stack trace.