Data Structures Advanced with Java - Regular Exam

Do not modify the interface or the package, or anything from the given resources. In Judge, you only upload the archive of the corresponding package.

3. MLM Service – 100pts

We've all heard of Multi Level Marketing. It's a system in which each level hires people underneath them and earns a commission for every sale made down the line. In this service, we'll be working with a Seller who has total earnings and a Set of people that they successfully hired. Each Seller is identified by their unique sellerId.

- **`void addSeller(Seller seller)`** - Adds a seller to the system. If the seller already exists - throw **IllegalArgumentException**
- **`void hire(Seller parent, Seller newHire)`** - Add the **newHire** to the collection of people that that parent hired. If the parent is missing or the newHire already exists - throw IllegalArgumentException
- **`boolean exist(Seller seller)`** - returns whether the **Seller** exists in our system.
- **`void fire(Seller seller)`** - Fire a seller from our system. When firing a seller all of the people that they hired should be transferred to the person that hired them.
  Example:
  Person A hired person B
  Person B hired person C & D
  If we fire B then C & D should be added as hires of A
- **`void makeSale(Seller seller, int amount)`** - making a sale means giving a commission to all people in the hire chain above. Each person above this seller receives 5% of the amount. What's left is added to that seller's earnings. There will always be enough in the amount for everybody in the chain. The seller will always be valid
  Example:
  Person A hires person B
  Person B hires person C
  If C makes a sale for 100 that means that
  B earns 5
  A earns 5
  C earns 90
- **`Collection<Seller> getByProfits()`** - return all Sellers orders by their profits from highest to lowest
- **`Collection<Seller> getByEmployeeCount()`** - return all Sellers by the count of people they have hired - highest to lowest. When calculating hires take into account all levels not just the people you directly hired. For people with the same count order by order of addition.
  Example:
  Person A hired person B
  Person B hired person C & D
  That means person A has 3 employees, person B - 2 employees, C & D - 0 employees
- **`Collection<Seller> getByTotalSalesMade()`** - return all sellers by the count of sales that each one made highest to lowest

4. MLM Service – Performance – 50pts

For this task, you will only be required to submit the **code from the previous problem**. If you are having a problem with this task you should **perform detailed algorithmic complexity analysis**, and try to **figure out weak** spots inside your implementation.

For this problem, it is important that other operations are **implemented correctly** according to the specific problems: **addSeller**, **hire**, **exist,** etc…

You can submit code to this problem **without full coverage** from the previous problem, **not all test cases** will be considered, only the **general behavior** will be important, **edge cases** will mostly be ignored such as throwing exceptions, etc…