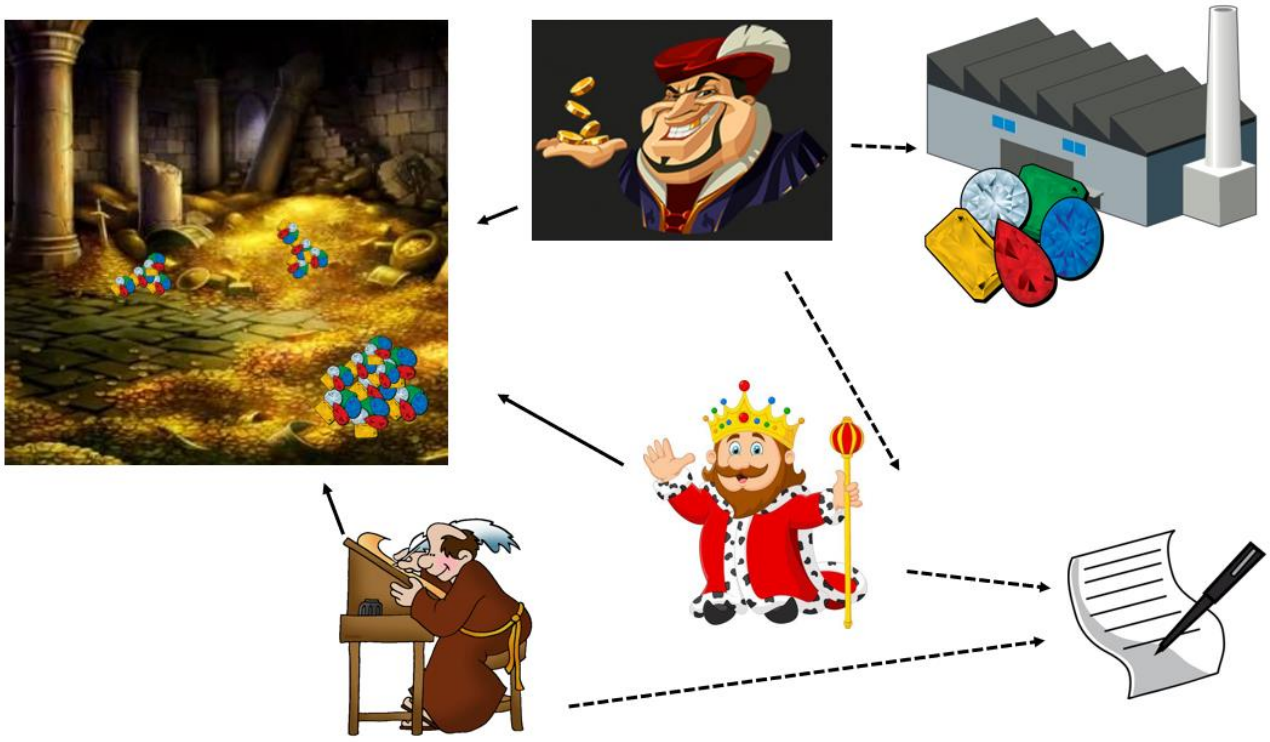
Assignment 3, SDJ2-S19
(Readers & Writers, Flyweight, Singleton)

The assignment:

You are going to implement a thread example based on the readers-writers problem.
The below image is a sketch class diagram:



This program will simulate a kingdom, where tax collectors collect valuables from the people, accountants count the valuables, and the king spends the valuables to hold parties.

You will need at least the following classes:

1 - ValuableFactory

The ValuableFactory. This is the factory class in the flyweight pattern. The flyweight objects in this assignment are of the type Valuable.
This Valuable is either an interface or an abstract class, that is up to you.
An example of a Valuable sub-class could be:

```
public class Diamond extends Valuable {
    @Override
    public String getName() { return "Diamond"; }

    @Override
    public int getValue() { return 25; }
}
```

The ValuableFactory is responsible for creating and caching Valuable objecst. Create a couple of different types, e.g.: Diamond, GoldCoin, Jewel, Ruby, WoodenCoin, Cow etc…

2 - TreasureRoom

The TreasureRoom contains a list of Valuables. It must control access to this list using a Readers/Writers-problem approach. Which approach is up to you. Pick one from class, or come up with your own.

It must at least have methods to add one Valuable to the list, to get a Valuable from the list and to remove and get a Valuable from the list.

3 - Accountant

The Accountant is a "reader" class. It implements Runnable, so it can be run in a separate thread. The accountant will have a while(true) loop in the run method.

- He will acquire read access
- Count the total sum of the Valuables worth in the list (it may include a sleep to simulate it takes time to count the Valuables)
- Print the total sum out
- Release read access
- Sleep for a little while

4 - TaxCollector

The TaxCollector is meant to ride around the country, and collect taxes for the king, in terms of Valuables. He will be given a valuable target, and must collect Valuables until their total worth meets this target.
This class is also Runnable, to be run in a thread.
The TaxCollector will do the following:

- Generate a random number, e.g. between 50 and 200.
- He will then a number of times get a random Valuable from the ValuableFactory (meaning he must know about possible Valuables), and between each retrieval of a Valuable sleep for a little while. This will continue, until he has a list of Valuables with a total worth equal to or more than the original target number.
- He will then acquire write access to the TreasureRoom
- Store all the collected Valuables (one at a time with a sleep in between to simulate that it takes time)
- Release write access.
- Sleep for a little while
- Start over.

5 - King

The king will occasionally hold a party, and to pay for this, he will retrieve valuables from the TreasureRoom.

- Similar to the TaxCollector, the King will generate a random number, e.g. 50-150, to pay for the next party.
- He will acquire write access
- Retrieve the valuables, and if the target worth cannot be met, he will cancel the party, and put the valuables back. (again it could include a sleep to simulate it takes time to get the desired valuables)
- Release write access

- If the target is met, he will hold a party.
- Sleep for a while
- Start over

6 - Catalog

The last class is logging functionality, to keep track of the income and outcome of the kingdom. Use the Singleton pattern here.
Use this logger-class to log out to the console, what happens in your program. E.g. when and how much a TaxCollector adds to the TreasureRoom, when the King holds a party or cancels it, when the Accountants has calculated the wealth in the TreasureRoom. You may also include logs when one of them are waiting to enter the TreasureRoom.

You may create other classes if needed.
Create a class with a main method to start the program. Create a bunch of Accountants, a couple of TaxCollectors and one King. Run the program and inspect the output from the Singleton-Logger.

Deadline

You can work on the assignment in the following SDJ2 sessions, – and of cause outside of class.
All sessions between the Monday the 6th of May until Friday the 10th of May.

Deadline: Sunday the 12th of May.

Format

You are allowed to work in groups, but you must each hand in a class diagram and the source files for all relevant Java classes, in a single zip-file.

Evaluation

Your hand-in will be registered **and counts for one of the exam requirements**. No formal feedback should be expected.