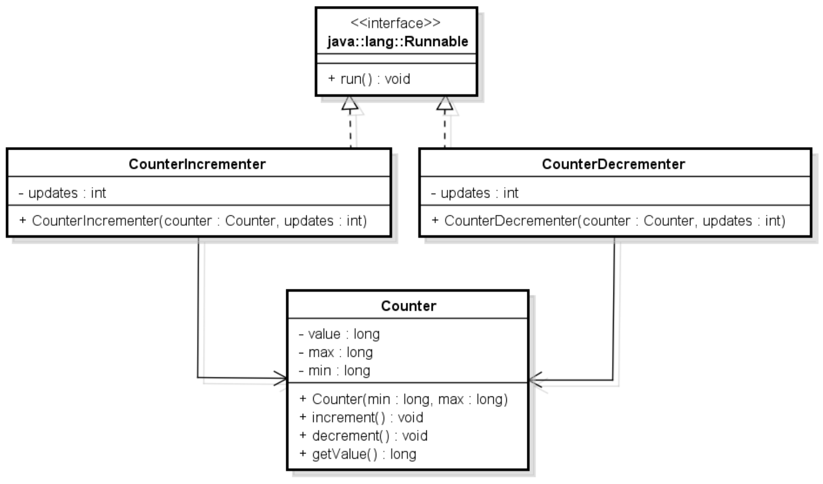
Exercises Session 4 SDJ2

Exercise 04.01

Implement the following system (see below)



A class `Counter` as a Monitor class (with private instance variables and all methods synchronized):

- A constructor setting value to 0 and min and max to whatever the values of the two arguments
- A method increment() incrementing the value by 1 (and let the calling thread wait if counter >= max)
- A method decrement() decrementing the value by 1 (and let the calling thread wait if counter <= min)
- A method getValue() returning the value

A class `CounterIncrementer` implementing `Runnable`. In the `run` method create a loop with `updates` loop cycles and call the `Counter` method `increment()` in the loop body. After the loop, print out the value of the counter. Class `CounterDecrementer` is almost the same, except that this one calls `decrement()`.

Implement a class with a `main` method in which you create a `Counter` object, pass this to 2 `CounterIncrementer` objects and 2 `CounterDecrementer` objects (all with the second argument set to 200, i.e. 200 updates), create 4 threads with each of the 4 Runnable objects and start up the 4 threads.

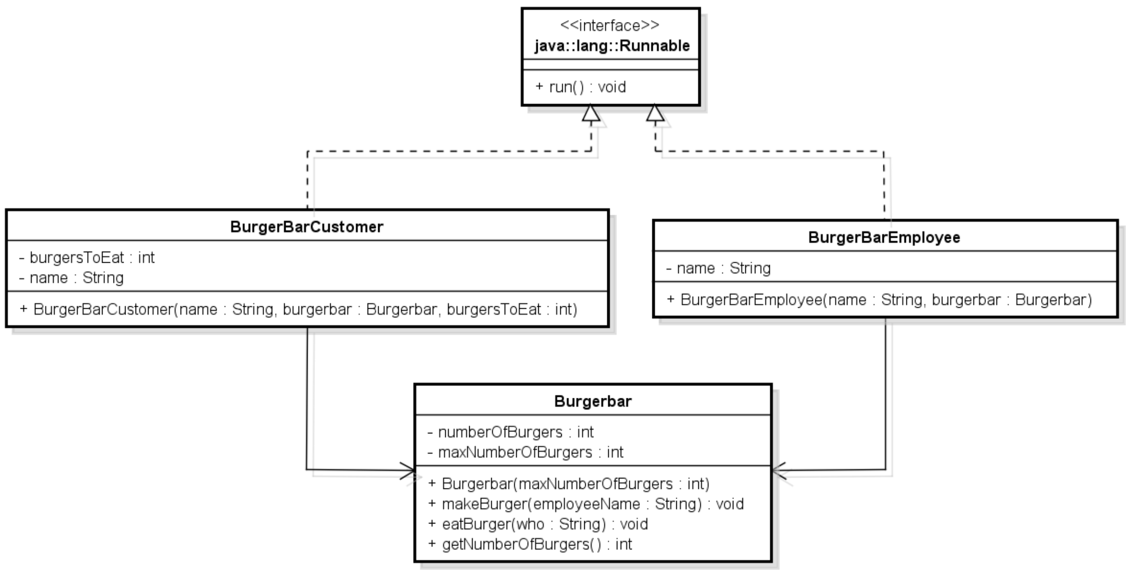
…insert a few print-statements in class `Counter` to see when it is being updated (and by which thread), e.g. insert something similar to the following when `value` is updated and when a thread is blocked:

```
System.out.println(value + ": " + Thread.currentThread().getName());
```

Run the program a few times and inspect the output.

Exercise 04.02

Implement the following system implementing a burger bar with customers and employees. Employees are making burgers and customers are eating burgers (see below)



<<interface>>
java::lang::Runnable
+ run() : void
BurgerBarCustomer
- burgersToEat : int
- name : String
+ BurgerBarCustomer(name : String, burgerbar : Burgerbar, burgersToEat : int)
BurgerBarEmployee
- name : String
+ BurgerBarEmployee(name : String, burgerbar : Burgerbar)
Burgerbar
- numberOfBurgers : int
- maxNumberOfBurgers : int
+ Burgerbar(maxNumberOfBurgers : int)
+ makeBurger(employeeName : String) : void
+ eatBurger(who : String) : void
+ getNumberOfBurgers() : int

A class `Burgerbar` as a Monitor class (with private instance variables and all methods synchronized):

- A constructor setting the number of burgers to 0 and maxNumberOfBurges to the values of the argument
- A method `makeBurger(…)` incrementing the number of burgers by 1 (and let the calling thread wait if counter >= maxNumberOfBurges)
- A method `eatBurger(…)` decrementing the number of burgers by 1 (and let the calling thread wait if counter <= 0)
- A method `getNumberOfBurgers()` returning the number of burgers

A class `BurgerbarEmployee` implementing `Runnable`. In the `run` method, create an infinite loop and call the method `makeBurger(…)` in the loop body. Use a sleep to simulate that it takes some time to make the burger (but not inside a synchronized method because sleep is not releasing the monitors lock).

A class `BurgerbarCustomer` implementing `Runnable`. In the `run` method, create a loop with `burgersToEat` loop cycles and call the method `eatBurger(…)` in the loop body. Use a sleep to simulate that it takes some time to eat the burger.

Implement a class with a `main` method in which you create a `BurgerBar` object, pass this to 2 `BurgerbarEmployee` objects and 5 `BurgerbarCustomer` objects (give values for parameters), create all 7 threads with each of the Runnable objects and start up all threads.

…insert a few print-statements in class `BurgerBar` to see when a burger is made and when it is eaten – and by whom, e.g. insert something similar to the following when `numberOfBurgers` is updated and when a thread is blocked:

```
      System.out.println(who + " is ready to eat a burger (" +
                                    numberOfBurgers + " left)");
```

Run the program a few times and inspect the output.
Extra: Try to close the burger bar when there are no more customers