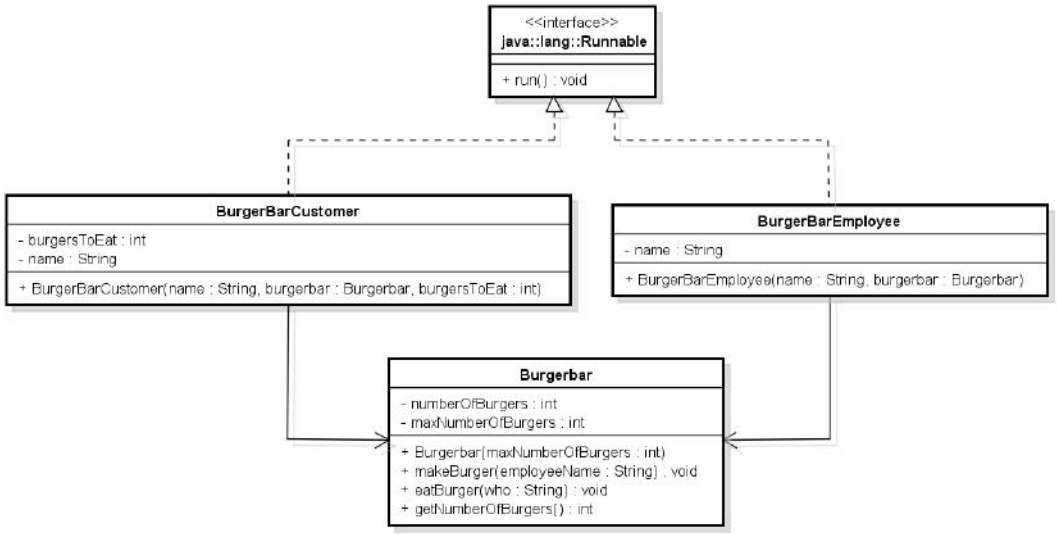
Exercise: Burgerbar

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



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 maxNumberOfBurgers to the values of the argument
- A method `makeBurger(…)` incrementing the number of burgers by 1 (and let the calling thread wait if counter >= maxNumberOfBurgers)
- 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 never 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 just before 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