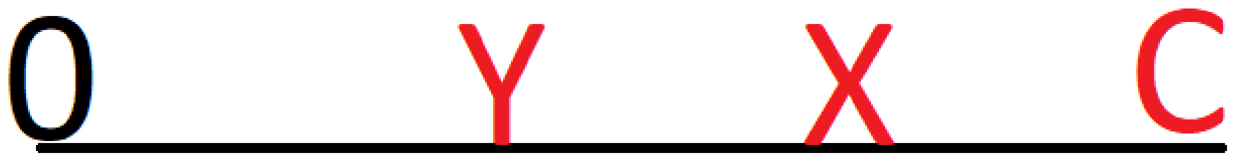
Problem summary
There are two bidders that bid for a mount of Product. Each time a couple of products are shown in a bid . Every time bidders know the last cash bidded to the product by both of them. We need to make a program to solve the problem

Algorithm.
After first bid the bidders will know the last cash bidded by both of them. Assume bidder 1 bids X and bidder 2 bids Y . Bidders have the same Cash limit C. We need to find an algorithm for bidder 1 to determine the next bid cash.

In case X<Y

Here we have four area 0-X , x-y , y-c
The next cash amount can be any number allocated in one of the four areas. My assumption is that area Y-C has higher probability than other areas. So the next cash amount should be between Y-C.Because bidder 1 will assume that bidder 2 next bid will be less or near Y.
But which number between Y-C should we choose. It should be the least number bigger than y that bidder 2 can choose. We do not know the bidder 2 next cash so here our algorithm has a parameter called step S . This will be multiplied each time bidder 1 loses.
For example
Next bid for X is y + S , if this bid does not win then the following bid will be y+ 2S , if not succeed then the following bid will be y+3S and so on.

In case X>Y



Here also we will assume bidder 2 will choose a number between X-C. so we will choose X+S

Analysis
So this algorithm will increase the bid based on parameter S and the algorithm will react according to the other bidder's last bid. This will increase the probability of winning.

Design of the program

We have Three main components .
BidController: This class runs the bid over Product quantity
Bidder: This interface represents the person who bids cash to get the product
Strategy: This interface for Classes that determines the next cash. Here we only implement one Strategy that we describe above.
ExponentialStrategy: This class implements Strategy interface according to our algorithm

Technology of the program

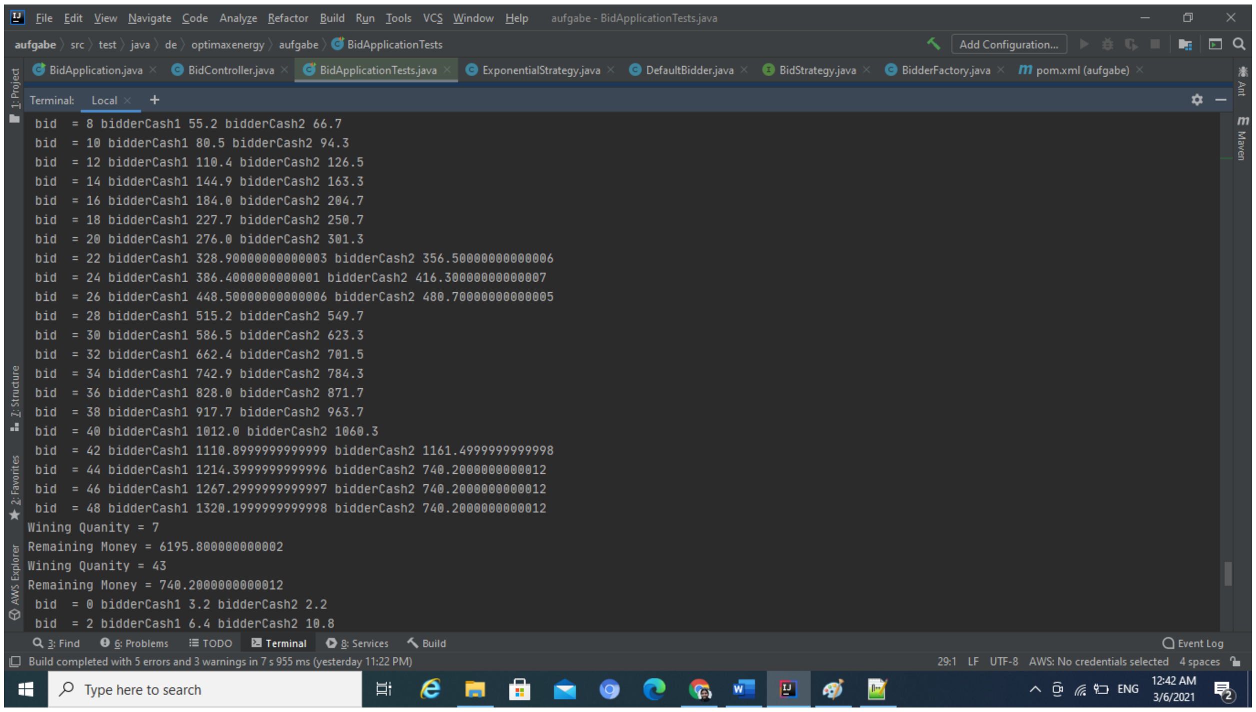
Here we use spring boot with maven and we create test cases in the Test folder to test the algorithm.

Running the program

Download the code from my github https://github.com/aymanElshayeb/optimax
run
mvnw clean install

This command will run the test cases which Test our algorithm
Open the test case under `de.optimaxenergy.aufgabe.BidApplicationTests`

Sample of the running



```
 bid  = 8 bidderCash1 55.2 bidderCash2 66.7
 bid  = 10 bidderCash1 80.5 bidderCash2 94.3
 bid  = 12 bidderCash1 110.4 bidderCash2 126.5
 bid  = 14 bidderCash1 144.9 bidderCash2 163.3
 bid  = 16 bidderCash1 184.0 bidderCash2 204.7
 bid  = 18 bidderCash1 227.7 bidderCash2 250.7
 bid  = 20 bidderCash1 276.0 bidderCash2 301.3
 bid  = 22 bidderCash1 328.90000000000003 bidderCash2 356.50000000000006
 bid  = 24 bidderCash1 386.4000000000001 bidderCash2 416.30000000000007
 bid  = 26 bidderCash1 448.50000000000006 bidderCash2 480.70000000000005
 bid  = 28 bidderCash1 515.2 bidderCash2 549.7
 bid  = 30 bidderCash1 586.5 bidderCash2 623.3
 bid  = 32 bidderCash1 662.4 bidderCash2 701.5
 bid  = 34 bidderCash1 742.9 bidderCash2 784.3
 bid  = 36 bidderCash1 828.0 bidderCash2 871.7
 bid  = 38 bidderCash1 917.7 bidderCash2 963.7
 bid  = 40 bidderCash1 1012.0 bidderCash2 1060.3
 bid  = 42 bidderCash1 1110.8999999999999 bidderCash2 1161.4999999999998
 bid  = 44 bidderCash1 1214.3999999999996 bidderCash2 740.2000000000012
 bid  = 46 bidderCash1 1267.2999999999997 bidderCash2 740.2000000000012
 bid  = 48 bidderCash1 1320.1999999999998 bidderCash2 740.2000000000012
Wining Quanity = 7
Remaining Money = 6195.800000000002
Wining Quanity = 43
Remaining Money = 740.2000000000012
 bid  = 0 bidderCash1 3.2 bidderCash2 2.2
 bid  = 2 bidderCash1 6.4 bidderCash2 10.8
```

Build completed with 5 errors and 3 warnings in 7 s 955 ms (yesterday 11:22 PM)

Test class

```
package de.optimaxenergy.aufgabe;

import de.optimaxenergy.aufgabe.bidder.Bidder;
import de.optimaxenergy.aufgabe.bidder.BidderFactory;
import de.optimaxenergy.aufgabe.bidder.strategy.ExponentialStrategy;
import de.optimaxenergy.aufgabe.controller.BidController;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.util.Assert;

@SpringBootTest
class BidApplicationTests {

   @Test
   void contextLoads() {
   }

   @Test
   void testBidderWithDifferentStep() {
       Bidder bidder1 = BidderFactory.getBidder(0, 10000, new
ExponentialStrategy(3.2));
       Bidder bidder2 = BidderFactory.getBidder(0, 10000, new
ExponentialStrategy(2.2));
       BidController bidController = new BidController(bidder1, bidder2, 50);
       bidController.run();
       bidder1.printStatus();
       bidder2.printStatus();
       Assert.isTrue(bidder1.getQuantity() == 10, "bidder1 winning quanity is
incorrect");
       Assert.isTrue(bidder2.getQuantity() == 40, "bidder2 winning quanity is
incorrect");
   }

   @Test
   void testBidderWithSameStep() {
       Bidder bidder1 = BidderFactory.getBidder(0, 10000, new
ExponentialStrategy(2.3));
       Bidder bidder2 = BidderFactory.getBidder(0, 10000, new
ExponentialStrategy(2.3));
       BidController bidController = new BidController(bidder1, bidder2, 50);
       bidController.run();
       bidder1.printStatus();
       bidder2.printStatus();
       Assert.isTrue(bidder1.getQuantity() == 7, "bidder1 winning quanity is
incorrect");
```

```
        Assert.isTrue(bidder2.getQuantity() == 43, "bidder2 winning quanity is
incorrect");
    }

}
```