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 so that bidder can win as much as he can of the product

Algorithm.

Assume bidder 1 and bidder 2 have the same cash limit C . The bidder 1 and The bidder 2 will make R round for product X the number of Round R = product number / 2

The Bidder 1 will target to win the half of the round so he will bid by constant amount 2 C/R . Bidder 2 now has two cases

Bidder 2 bids with value < 2C/R

this means Bidder 1 will win the half of the round because he can afford more than Bidder 1

Bidder 2 bids with value > 2C/R

This means Bidder 2 wins 1 bid but he can not afford to win the remaining bid to reach the half of total bids and in this case Bidder 1 can reduce the bid value to less than 2C/R because bidder 2 can not afford 2C/R any more to win the half.

Each time Bidder 2 bids with value >2C/R , bidder 1 reduces the bid value because bidder 2 can not afford to reach the half . After bidder 2 loses all his money bidder 1 will bid by less possible value to win the remaining bids.

The value that bidder 1 can afford each time Bidder 2 bids higher value of 2C/R equals Remaining cash of bidder 2 / (half of the Round - Winning round by bidder2)

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.

halfRoundsStrategy: This class implements Strategy interface according to our algorithm

IncreamentalWithStepStrategy: This class implements another strategy to test against the HalfRound Algorithm. The strategy starts with base step and increase this step incremental until winning first bid

ConstantStrategy: This class implements another strategy to test against the HalfRound strategy. The strategy bid allows a constant number until the end of cash.

Design Pattern:

We use the strategy design pattern to change between different algorithms and we use Factory design patterns to create bidders.

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  = 980 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 981 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 982 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 983 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 984 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 985 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 986 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 987 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 988 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 989 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 990 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 991 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 992 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 993 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 994 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 995 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 996 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 997 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 998 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 999 bidderCash1 0.0035460992907801418 bidderCash2 0.0
bid  = 1000 bidderCash1 0.0035460992907801418 bidderCash2 0.0
Wining Quanity = 1566
Remaining Money = 99796.82414474015
Wining Quanity = 434
Remaining Money = 0.0
```

Test class

```
package de.optimaxenergy.aufgabe;

import de.optimaxenergy.aufgabe.bidder.Bidder;
import de.optimaxenergy.aufgabe.bidder.BidderFactory;
import de.optimaxenergy.aufgabe.bidder.strategy.ConstantStrategy;
import de.optimaxenergy.aufgabe.bidder.strategy.HalfRoundsStrategy;
import de.optimaxenergy.aufgabe.bidder.strategy.IncreamentalWithStepStrategy;
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 testHalfRoundVsConstantBidders() {
        double cashLimit = 100000;
        int productNumber = 2000;
        int numberOfbid = Math.round(productNumber/2);
         Bidder bidder1 = BidderFactory.getBidder(0, cashLimit, new
HalfRoundsStrategy(cashLimit, numberOfbid));
         Bidder bidder2 = BidderFactory.getBidder(0, cashLimit, new
ConstantStrategy(3 *(cashLimit/numberOfbid) ));
         BidController bidController = new BidController(bidder1, bidder2,
productNumber);
         bidController.run();
         bidder1.printStatus();
         bidder2.printStatus();
         Assert.isTrue(bidder1.getQuantity() == 1332, "bidder1 winning quanity is
incorrect");
         Assert.isTrue(bidder2.getQuantity() == 668, "bidder2 winning quanity is
incorrect");
    }
    @Test
    void testHalfRoundsVsIncrementalWithStepBidders() {
         double cashLimit = 100000;
         int productNumber = 2000;
         int numberOfbid = Math.round(productNumber/2);
         Bidder bidder1 = BidderFactory.getBidder(0, cashLimit, new
HalfRoundsStrategy(cashLimit, numberOfbid));
```

```
        Bidder bidder2 = BidderFactory.getBidder(0, cashLimit, new
IncreamentalWithStepStrategy(1.2) );
        BidController bidController = new BidController(bidder1, bidder2,
productNumber);
        bidController.run();
        bidder1.printStatus();
        bidder2.printStatus();
        Assert.isTrue(bidder1.getQuantity() == 1566, "bidder1 winning quanity is
incorrect");
        Assert.isTrue(bidder2.getQuantity() == 434, "bidder2 winning quanity is
incorrect");
    }}
```