```
//package Assingment;

import java.io.File;
import java.util.ArrayList;
import java.util.Scanner;

public class OnlineAuctionSystem {

    // Online Auction system that stores multiple Auctions/ bidders
    // Create Auction / create Bidder /
    // helps bidders place bid on the lots
    // calculates the fees owed by all the bidder to the system


    // Declaration of variable to store
    private ArrayList<Auction> auctionsList = new ArrayList<>();      //Arraylist of
 auctions to store multiple auctions
    private ArrayList<Bidder> bidders = new ArrayList<>();            // Array list
of bidder to store multiple Bidders


    // Constructor to create a new auction method with check constraints
    // of bad data and Redundant lot numbers.
    public Auction createAuction(String name, int firstLot, int lastLot, int minBid)
{

        boolean noRedundantLots = true;
        boolean noBadData    = true;

        noRedundantLots = checkRedundantLots(firstLot,lastLot,name);   // Check to s
ee if Redundant lot exists
        noBadData = noBadData(name,firstLot,lastLot,minBid);           // check to i
f bad data is sent for the user

        // if it's the first auction entered skip the redundant lot check
        if (auctionsList.size() == 0 && noBadData){
            Auction a1 = new Auction(name,firstLot,lastLot,minBid);    // Create a n
ew auction
            auctionsList.add(a1);                                      // Adding to
auction list
            return a1;
        }

        else if(auctionsList.size() > 0 && noRedundantLots && noBadData) {
            Auction a1 = new Auction(name,firstLot,lastLot,minBid);    // create a n
ew auction object
            auctionsList.add(a1);                                      // add object
 to auction list
            return a1;
        }

        else {
            return null;                                              // Return Null
 in other cases
        }

    }

    // Function that creates a new bidder and return a bidder object.
    // adds the new bidder object to the Bidders list
    public Bidder createBidder(String name){

        // check for bad data if null or empty string is passed
        if ((name == null) || name.isEmpty()){
            return null;                                             // Return null
in any case
        }
        else{
            int id = bidders.size()+1;                               // Generate bid
der id's for new bidder
            Bidder b1 = new Bidder(id,name);                         // Create a new
 with the name and id
```

```
            this.bidders.add(b1);                                     // Add bidder t
o the bidders list
            return b1;                                                // Return bidde
r
        }

    }

    // Returns the auction status of each auction
    // return a staring with Auction name / status [new/open/closed] / total of winn
ing bids from all lot of an auction
    public String auctionStatus(){

        String auctionString = "";               // initializing the string

        // loop through auction list and get name, status and total bid sum
        for (int i=0; i<auctionsList.size(); i++){
            auctionString += auctionsList.get(i).getName()+"\t"
                            +auctionsList.get(i).getStatus()+"\t"
                            +auctionsList.get(i).getTotalBidSum()+"\n";
        }

        return auctionString;                     // returns the appended string
    }

    // load bids automatically from a given file name
    // return the number of bids placed successfully
    public int loadBids(String fileName) {

        // variable to store extract from the file
        int numberOfValidBids = 0;
        String stringLine="";
        String[] splitString;
        int bidder ;
        int lotNo;
        int bidAmount;
        int status ;
        int sucessBids =0;

        // try to process bids from the lot
        try{

            File file = new File(fileName);
            Scanner sc = new Scanner(file);

            // loop to read each line of the file until the end of file
            while (sc.hasNextLine()){

                stringLine = sc.nextLine();                        // get the line of
 string from the file
                splitString = stringLine.split("\t");       // split files based on
"\t" spaces

                bidder = Integer.parseInt(splitString[0]);
                lotNo = Integer.parseInt(splitString[1]);
                bidAmount = Integer.parseInt(splitString[2]);


                status = placeBid(lotNo, bidder, bidAmount);

                if((status == 2)|| (status == 3) || (status == 4)){
                    sucessBids++;                                   // increment the
number of bids if it was a valid bid
                }

            }

            return sucessBids;                                     // Return the numb
er of successfully returned bids
        }
        catch (Exception e){
            return sucessBids;
        }
    }


    // Function that placed bids on a particular lot requested by the bidder
    public int placeBid(int lotNumber,Integer bidderId, int bid ){

        Bid newBid = new Bid(lotNumber,bidderId,bid);
// create a new bid
        int status = 1;
        boolean bidderPresent = false;

        bidderPresent = checkBidderIDExists(bidderId);
// check if the bidder exists in the system


        //check if the bidder id exits
        if(bidderPresent && bid > 0){

            // check for the Auction item in the Arraylist which contains the LOT nu
mber requested
            for(int i=0; i< auctionsList.size(); i++){

                // check which auction number it belongs to
                if(auctionsList.get(i).getFirstLot() <= lotNumber && lotNumber <= au
ctionsList.get(i).getLastLot()){

                    //send the bid to be placed on the lot under the auction which i
t is present.
                    if(auctionsList.get(i).getStatus() == "open"){
                        status = auctionsList.get(i).placeBid(newBid);
                    }
                }
            }
        }
         return status;
    }

    // check to see if the Bidder exists in the system
    // return true if the bidder exits
    public boolean checkBidderIDExists(int bidderId){

        boolean status = false;

        for (int i=0; i<this.bidders.size(); i++){
            if(this.bidders.get(i).getBidderId() == bidderId){
                status = true;
            }
        }

        return status;
    }

    // Function that calculates the fees owed by all the bidder on the winning aucti
on
    // returns the string that with bidder name / number of lots won / total amount
owed
    public String feesOwed() {

        String bidderString = "";

        // function the clears the bidder data before calculating
        clearBidderdata();

        // function that calculates the total winning bids of lots won for each bidd
er
        caluculateFeesOwed();

        for(int i=0; i<bidders.size(); i++){
            bidderString += bidders.get(i).getName()+"\t"
                    +bidders.get(i).getNumberOfLotsWon()+"\t"
                    +bidders.get(i).getTotolAmountOwed()+"\n";
        }
        return bidderString;

    }

    // calculate the Fees owed by each bidder on closed auction
    public void caluculateFeesOwed(){

        for(int i =0; i < auctionsList.size(); i++){

            // get only for the auctions that are closed.
            if(auctionsList.get(i).getStatus() == "closed"){

                // loop for all the lots in the section
                for(int j =0; j < this.auctionsList.get(i).getLots().size(); j++){

                    // update bidder id and the official bid for a lot in a closed a
uction
                    updateBidderData(auctionsList.get(i).getLots().get(j).getBidderI
D(), auctionsList.get(i).getLots().get(j).getOfficialBid());
                }

            }
        }
    }


    // function to set data into the bidders array list with the latest amounts
    public void updateBidderData(int bidderId, int officialBid){

    // loop through the bidders and update the bidder data
        for(int i =0; i < this.bidders.size(); i++){
            // if the bidder id matches bidder bidder
            if(this.bidders.get(i).getBidderId() == bidderId){

                int lotswon = bidders.get(i).getNumberOfLotsWon();
                int totalAmountOwed = bidders.get(i).getTotolAmountOwed();

                // update with new bidder data
                lotswon++;                                              // incre
ment the lots won.
                totalAmountOwed = totalAmountOwed + officialBid;            // incre
ment the total amount owed

                // update the same in the array list
                this.bidders.get(i).setNumberOfLotsWon(lotswon);
                this.bidders.get(i).setTotolAmountOwed(totalAmountOwed);
            }
        }

    }


    // Clear bidder data and set them to zero.
    public void clearBidderdata(){

        // loop through and set bidder lots won and total amount owed to zero
        for(int i =0; i < this.bidders.size(); i++){
            this.bidders.get(i).setNumberOfLotsWon(0);
            this.bidders.get(i).setTotolAmountOwed(0);
        }

    }


    // Function to check if Redundant lot is requested.
    // return false if there is an existing lot or true if not
    private boolean checkRedundantLots(int firstLot, int lastLot, String name){

        boolean check = true;

        for (int i=0; i < auctionsList.size(); i++){
            if((auctionsList.get(i).getFirstLot() <= firstLot && firstLot <= auction
sList.get(i).getLastLot() )
                    ||(auctionsList.get(i).getFirstLot() <= lastLot && lastLot <= au
ctionsList.get(i).getLastLot() )){

                // Make the check false if redundant lot exists
                check = false;
            }

            if(auctionsList.get(i).getName().equals(name)){
                // Make the check false if auction of the same name exits
                check = false;
            }
        }
        return check;
    }


    // check if bad data exists
    // and returns true where there is no bad data else false
    private boolean noBadData(String name, int firstLot, int lastLot, int minBid){
        boolean check = true;

        if((name==null) || (firstLot <=0) || (lastLot<=0) || (minBid <= 0) || name.i
sEmpty() || (firstLot > lastLot)){
            check = false;
        }
        return check;
    }


}
```