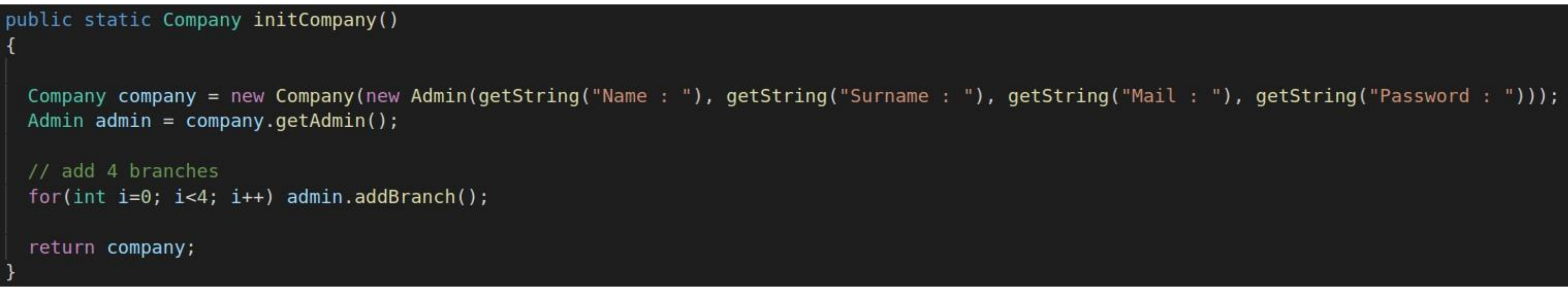
GTU Department of Computer EngineeringCSE 222/505 - Spring 2021 Homework 3 Report

Barış Ayyıldız 1901042252

1 SYSTEM REQUIREMENTS

At first, we should initialize a company.



```
public static Company initCompany()
{

  Company company = new Company(new Admin(getString("Name : "), getString("Surname : "), getString("Mail : "), getString("Password : ")));
  Admin admin = company.getAdmin();

  // add 4 branches
  for(int i=0; i<4; i++) admin.addBranch();

  return company;
}
```

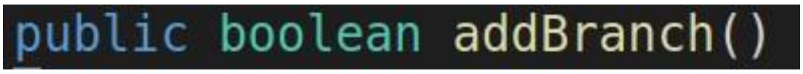
Company Constructor takes an Admin object. Admin constructor takesname,

surname, email and and password as parameters. In this method I use addBranch

method to create 4 branches initially.

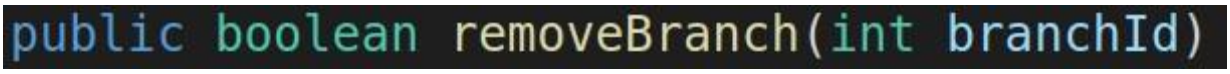
Admin

Admins can add and delete branch/employee/customers.



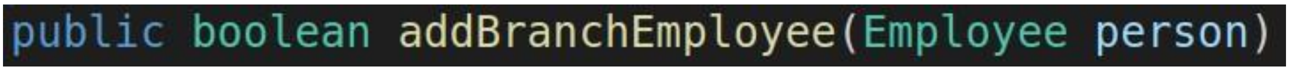
```
public boolean addBranch()
```

When deleting a branch, branch id is required. When adding, software creates aunique id so it is not required



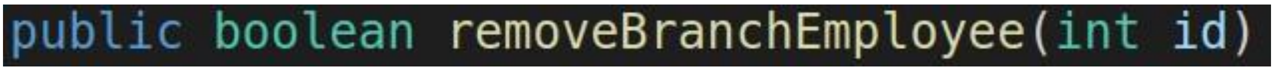
```
public boolean removeBranch(int branchId)
```

When adding a branch employee, an Employee object need to be passed.



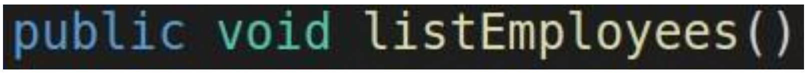
```
public boolean addBranchEmployee(Employee person)
```

When deleting it is only required to pass an employee id

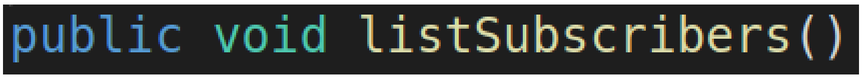


```
public boolean removeBranchEmployee(int id)
```

Admin also can list all the employees or all the subscribers(customers)



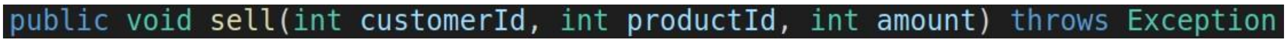
```
public void listEmployees()
```



```
public void listSubscribers()
```

Employee

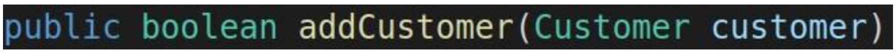
Branch employees can make in-shop sales. sales method takes customerId, productId and amount of products as parameters



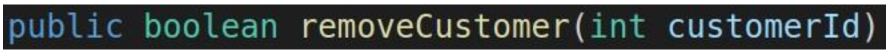
```
public void sell(int customerId, int productId, int amount) throws Exception
```

Company Members

Both Admin and Employee class inherits from CompanyMembers class. They both have access to add and remove a customer.

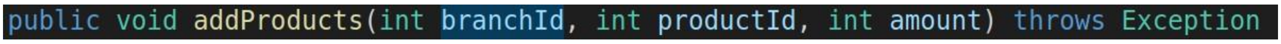


```
public boolean addCustomer(Customer customer)
```

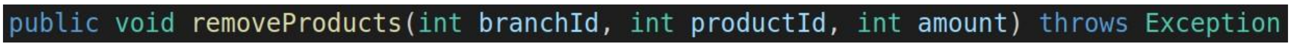


```
public boolean removeCustomer(int customerId)
```

Add and remove products

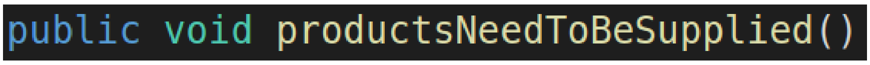


```
public void addProducts(int branchId, int productId, int amount) throws Exception
```



```
public void removeProducts(int branchId, int productId, int amount) throws Exception
```

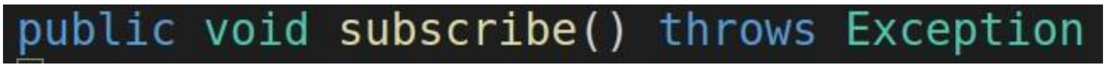
And list the products from all the branches that is out of stock



```
public void productsNeedToBeSupplied()
```

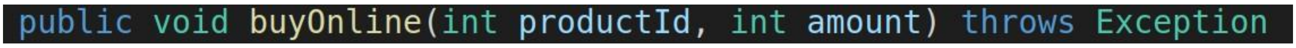
Customer

User is able to subscribe to the company after creating a customer object.

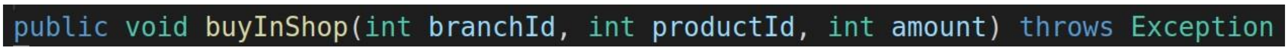


```
public void subscribe() throws Exception
```

This method sets customer object’s isSubscribed property to true. And sets the id to a unique integer value. Customer is able to buy online or buy in shop.

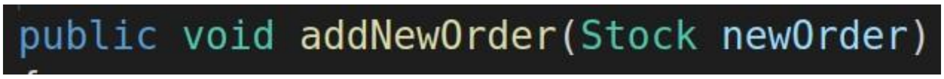


```
public void buyOnline(int productId, int amount) throws Exception
```



```
public void buyInShop(int branchId, int productId, int amount) throws Exception
```

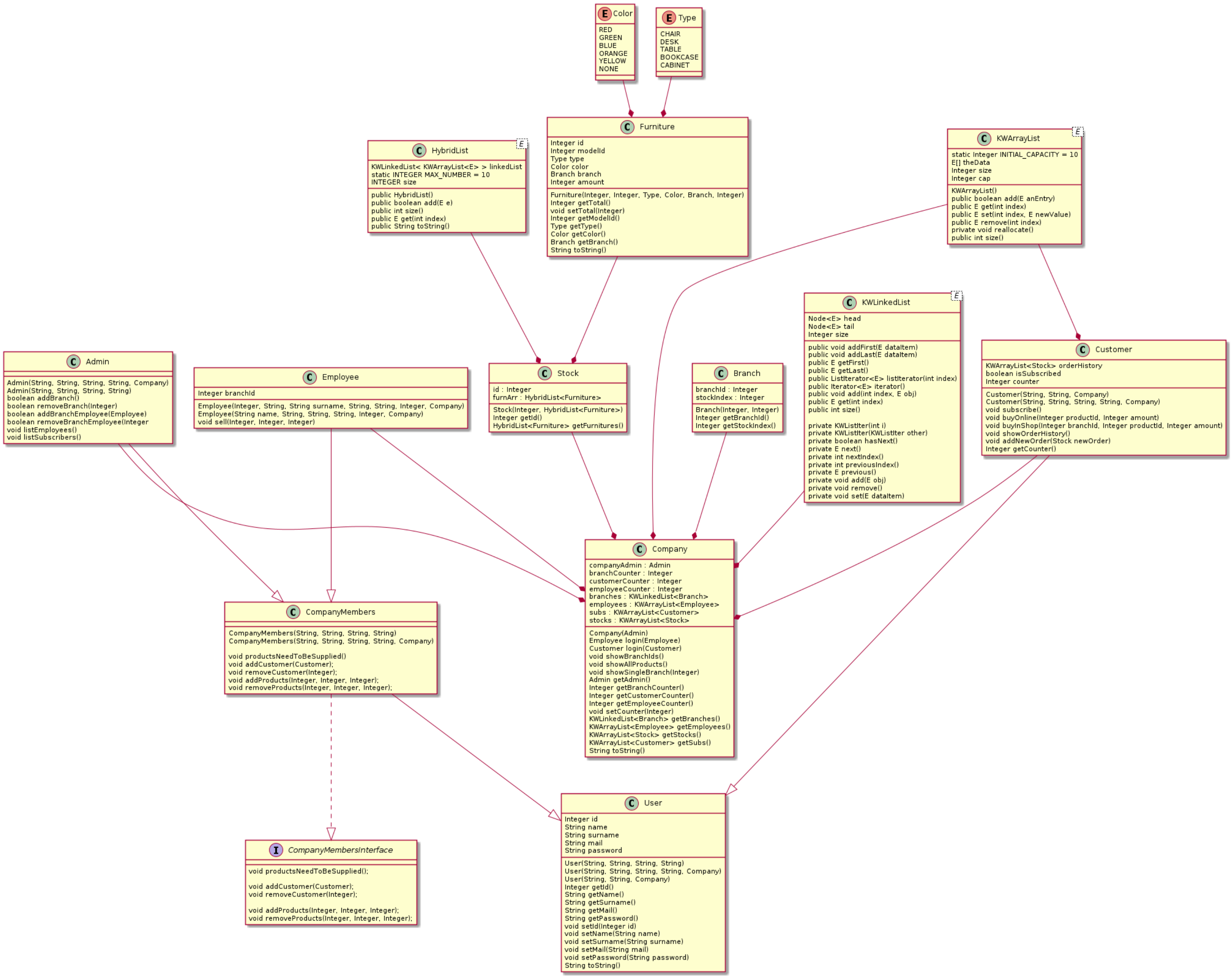
When customer buys online, they only pass productId and amount of products as a parameter. When buying in shop, branch id needs to be passed as a parameter also. After these methods, new order is inserted to orderHistory array. This array holds all the previous orders of a customer.



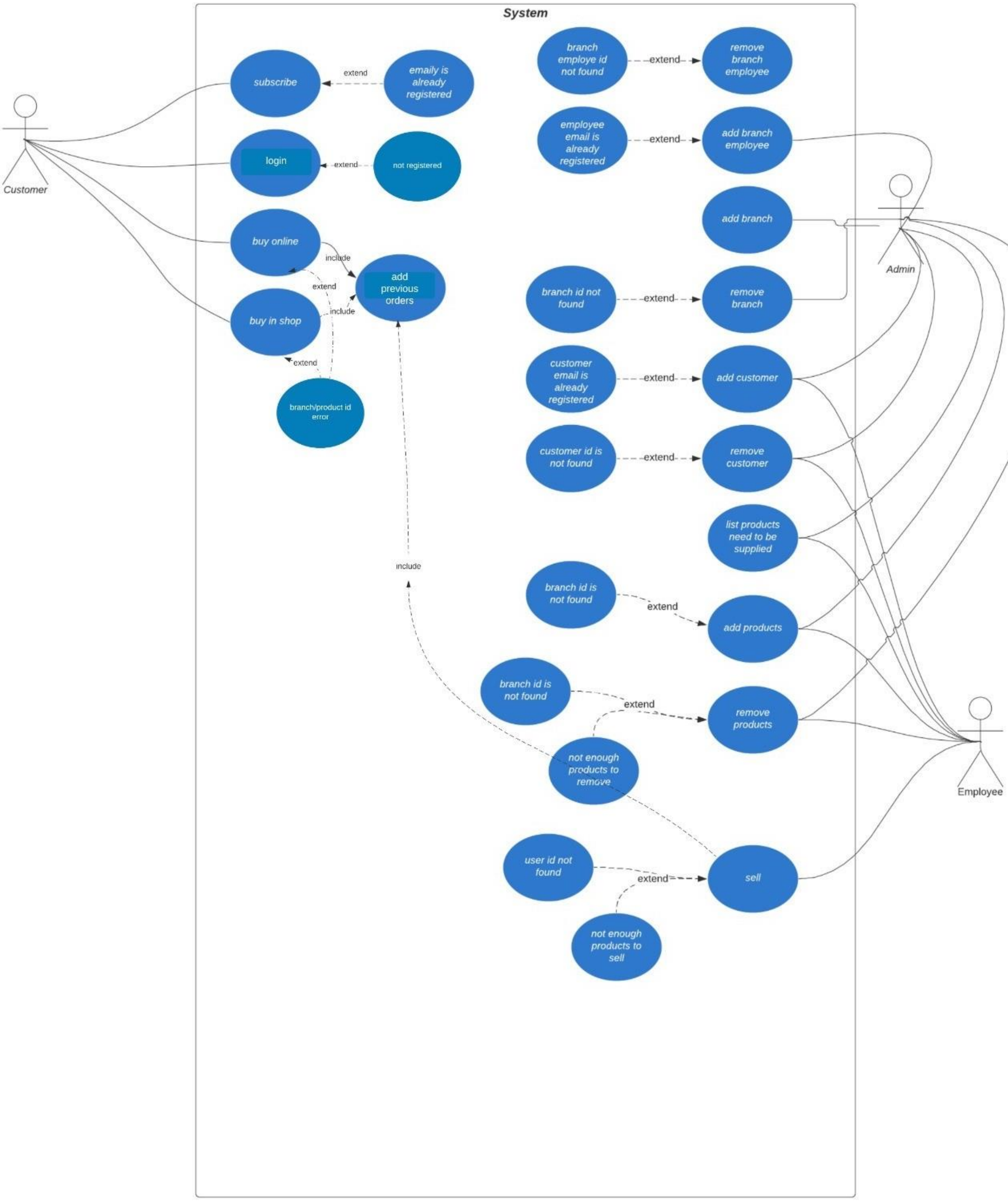
```
public void addNewOrder(Stock newOrder)
```

A stock object needs to be send as a parameter.

2 USE CASE AND CLASS DIAGRAMS



E Color
RED
GREEN
BLUE
ORANGE
YELLOW
NONE
E Type
CHAIR
DESK
TABLE
BOOKCASE
CABINET
C Furniture
Integer id
Integer modelId
Type type
Color color
Branch branch
Integer amount
Furniture(Integer, Integer, Type, Color, Branch, Integer)
Integer getTotal()
void setTotal(Integer)
Integer getModelId()
Type getType()
Color getColor()
Branch getBranch()
String toString()
C HybridList
E
KWLinkedList< KWArrayList<E> > linkedList
static INTEGER MAX_NUMBER = 10
INTEGER size
public HybridList()
public boolean add(E e)
public int size()
public E get(int index)
public String toString()
C KWArrayList
E
static Integer INITIAL_CAPACITY = 10
E[] theData
Integer size
Integer cap
KWArrayList()
public boolean add(E anEntry)
public E get(int index)
public E set(int index, E newValue)
public E remove(int index)
private void reallocate()
public int size()
C KWLinkedList
E
Node<E> head
Node<E> tail
Integer size
public void addFirst(E dataItem)
public void addLast(E dataItem)
public E getFirst()
public E getLast()
public ListIterator<E> listIterator(int index)
public Iterator<E> iterator()
public void add(int index, E obj)
public E get(int index)
public int size()
private KWListIter(int i)
private KWListIter(KWListIter other)
private boolean hasNext()
private E next()
private int nextIndex()
private int previousIndex()
private E previous()
private void add(E obj)
private void remove()
private void set(E dataItem)
C Admin
Admin(String, String, String, String, Company)
Admin(String, String, String, String)
boolean addBranch()
boolean removeBranch(Integer)
boolean addBranchEmployee(Employee)
boolean removeBranchEmployee(Integer
void listEmployees()
void listSubscribers()
C Employee
Integer branchId
Employee(Integer, String, String surname, String, String, Integer, Company)
Employee(String name, String, String, String, Integer, Company)
void sell(Integer, Integer, Integer)
C Stock
id : Integer
furnArr : HybridList<Furniture>
Stock(Integer, HybridList<Furniture>)
Integer getId()
HybridList<Furniture> getFurnitures()
C Branch
branchId : Integer
stockIndex : Integer
Branch(Integer, Integer)
Integer getBranchId()
Integer getStockIndex()
C Customer
KWArrayList<Stock> orderHistory
boolean isSubscribed
Integer counter
Customer(String, String, Company)
Customer(String, String, String, String, Company)
void subscribe()
void buyOnline(Integer productId, Integer amount)
void buyInShop(Integer branchId, Integer productId, Integer amount)
void showOrderHistory()
void addNewOrder(Stock newOrder)
Integer getCounter()
C Company
companyAdmin : Admin
branchCounter : Integer
customerCounter : Integer
employeeCounter : Integer
branches : KWLinkedList<Branch>
employees : KWArrayList<Employee>
subs : KWArrayList<Customer>
stocks : KWArrayList<Stock>
Company(Admin)
Employee login(Employee)
Customer login(Customer)
void showBranchIds()
void showAllProducts()
void showSingleBranch(Integer)
Admin getAdmin()
Integer getBranchCounter()
Integer getCustomerCounter()
Integer getEmployeeCounter()
void setCounter(Integer)
KWLinkedList<Branch> getBranches()
KWArrayList<Employee> getEmployees()
KWArrayList<Stock> getStocks()
KWArrayList<Customer> getSubs()
String toString()
C CompanyMembers
CompanyMembers(String, String, String, String)
CompanyMembers(String, String, String, String, Company)
void productsNeedToBeSupplied()
void addCustomer(Customer);
void removeCustomer(Integer);
void addProducts(Integer, Integer, Integer);
void removeProducts(Integer, Integer, Integer);
I CompanyMembersInterface
void productsNeedToBeSupplied();
void addCustomer(Customer);
void removeCustomer(Integer);
void addProducts(Integer, Integer, Integer);
void removeProducts(Integer, Integer, Integer);
C User
Integer id
String name
String surname
String mail
String password
User(String, String, String, String)
User(String, String, String, String, Company)
User(String, String, Company)
Integer getId()
String getName()
String getSurname()
String getMail()
String getPassword()
void setId(Integer id)
void setName(String name)
void setSurname(String surname)
void setMail(String mail)
void setPassword(String password)
String toString()


System
Customer
Admin
Employee
subscribe
extend
emaily is already registered
login
extend
not registered
buy online
include
add previous orders
extend
buy in shop
include
extend
branch/product id error
include
branch employe id not found
extend
remove branch employee
employee email is already registered
extend
add branch employee
add branch
branch id not found
extend
remove branch
customer email is already registered
extend
add customer
customer id is not found
extend
remove customer
list products need to be supplied
branch id is not found
extend
add products
branch id is not found
extend
remove products
not enough products to remove
user id not found
extend
sell
not enough products to sell

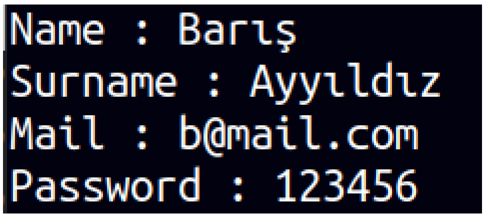
3 PROBLEM SOLUTION APPROACH

I have created a ListInterface<T> interface and List<T> class that implements it. Since we were not allowed to use any data structures other than arrays,I developed my own ArrayList like data structure. And it made the whole structure

very clean.

4 TEST CASES

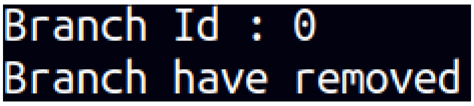
Create a company with administrator



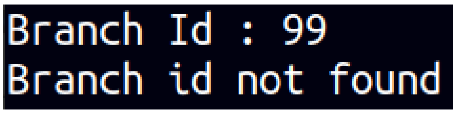
```
Name : Barış
Surname : Ayyıldız
Mail : b@mail.com
Password : 123456
```

Admin

Remove branch

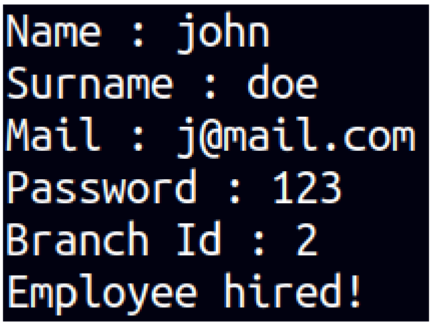


```
Branch Id : 0
Branch have removed
```

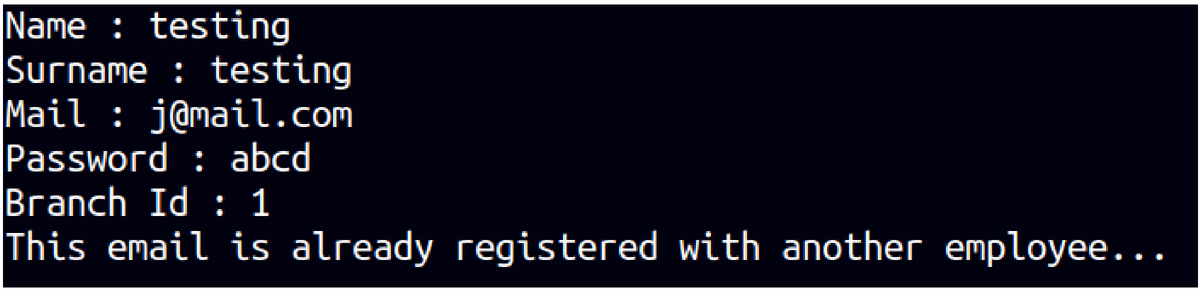


```
Branch Id : 99
Branch id not found
```

Add branch employee

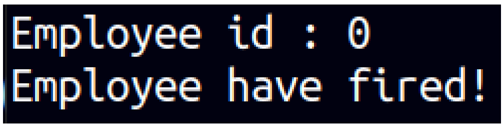


```
Name : john
Surname : doe
Mail : j@mail.com
Password : 123
Branch Id : 2
Employee hired!
```

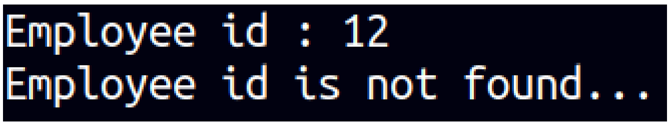


```
Name : testing
Surname : testing
Mail : j@mail.com
Password : abcd
Branch Id : 1
This email is already registered with another employee...
```

Remove branch employee

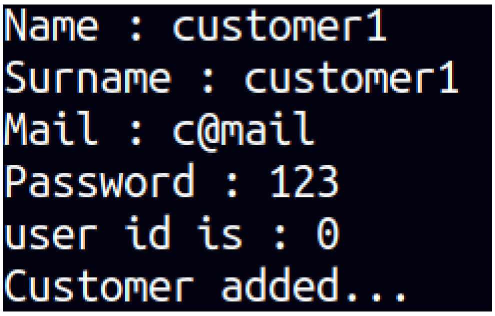


```
Employee id : 0
Employee have fired!
```

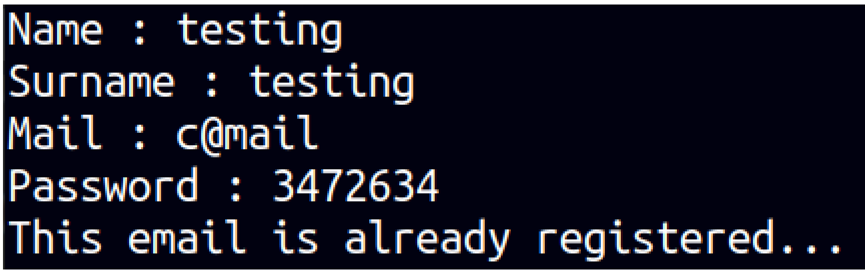


```
Employee id : 12
Employee id is not found...
```

Add customer

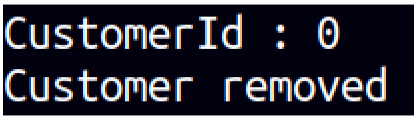


```
Name : customer1
Surname : customer1
Mail : c@mail
Password : 123
user id is : 0
Customer added...
```

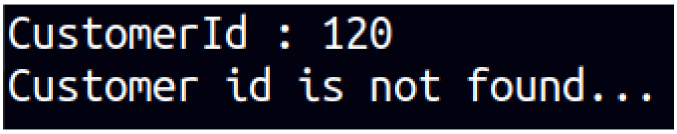


```
Name : testing
Surname : testing
Mail : c@mail
Password : 3472634
This email is already registered...
```

Remove customer

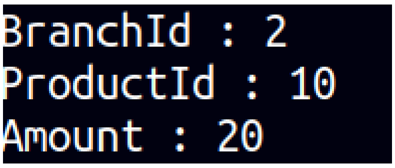


```
CustomerId : 0
Customer removed
```

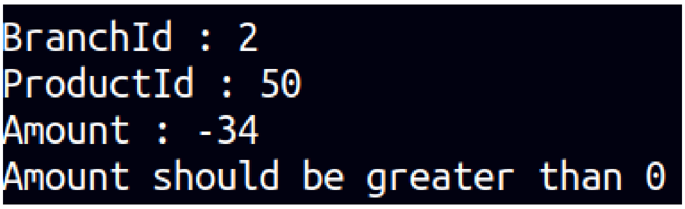


```
CustomerId : 120
Customer id is not found...
```

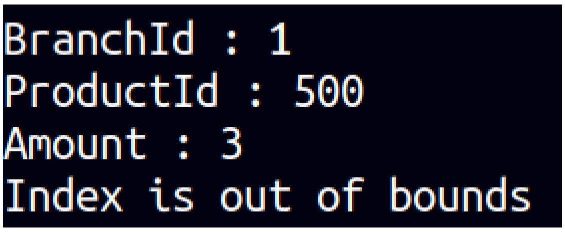
Add products



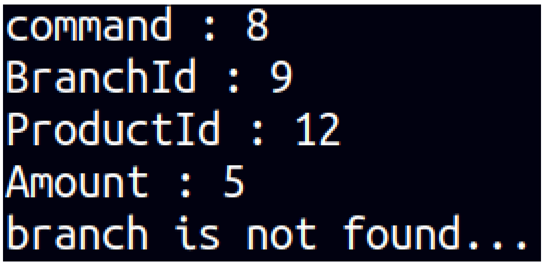
```
BranchId : 2
ProductId : 10
Amount : 20
```



```
BranchId : 2
ProductId : 50
Amount : -34
Amount should be greater than 0
```

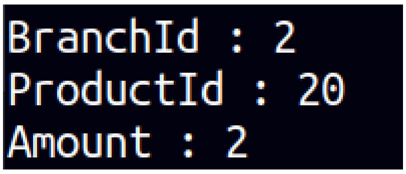


```
BranchId : 1
ProductId : 500
Amount : 3
Index is out of bounds
```

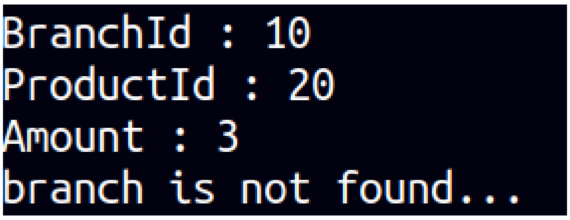


```
command : 8
BranchId : 9
ProductId : 12
Amount : 5
branch is not found...
```

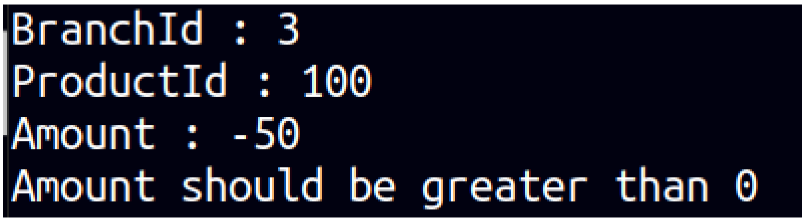
Remove products



```
BranchId : 2
ProductId : 20
Amount : 2
```



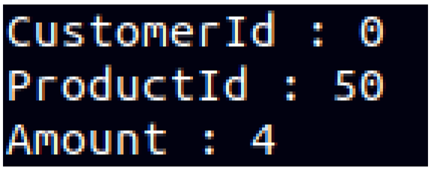
```
BranchId : 10
ProductId : 20
Amount : 3
branch is not found...
```



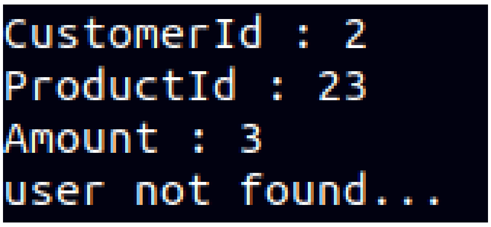
```
BranchId : 3
ProductId : 100
Amount : -50
Amount should be greater than 0
```

Employee

Sell



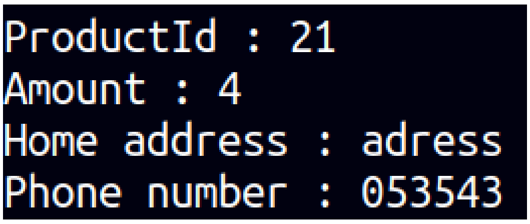
```
CustomerId : 0
ProductId : 50
Amount : 4
```



```
CustomerId : 2
ProductId : 23
Amount : 3
user not found...
```

Customer

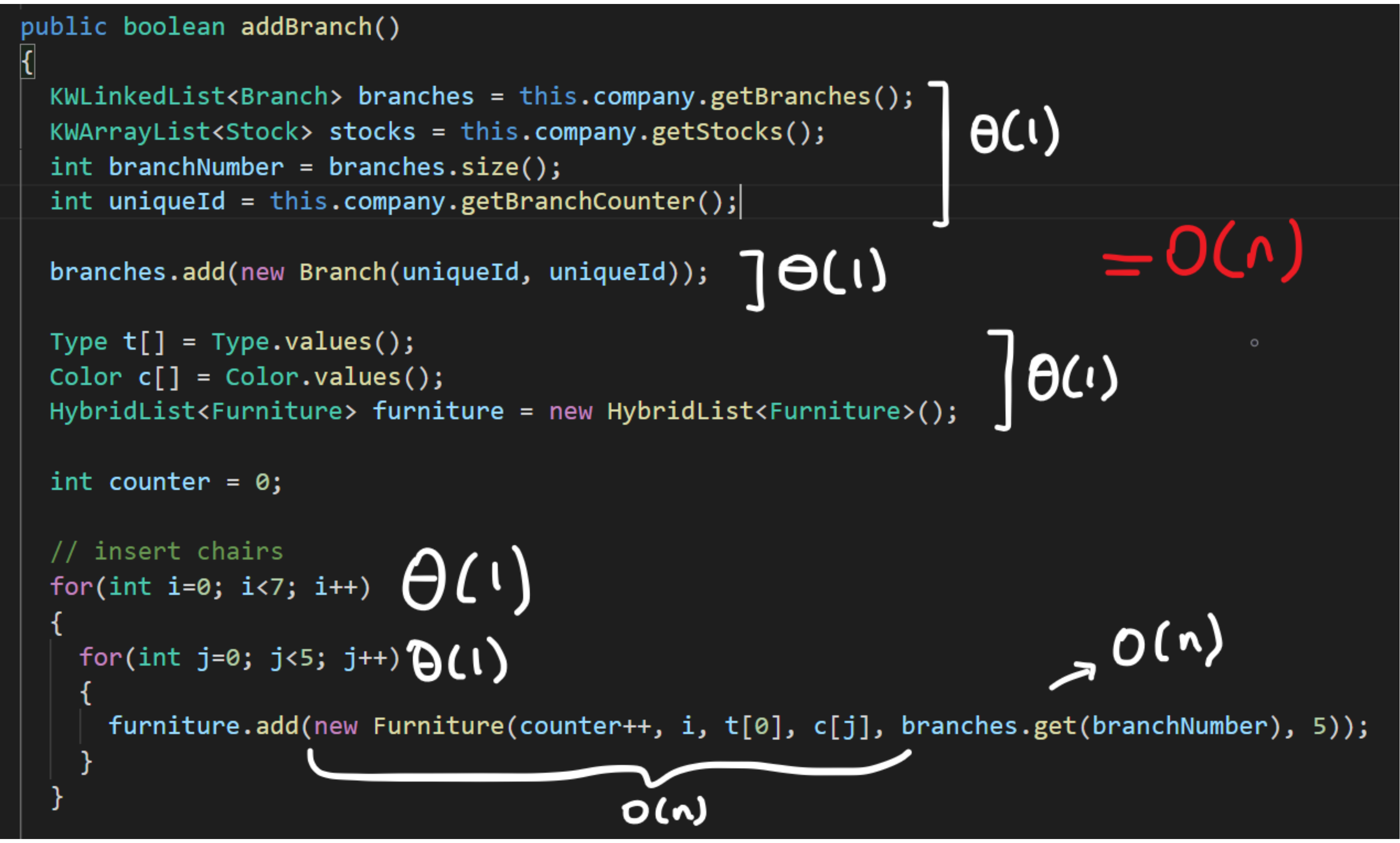
Buy online



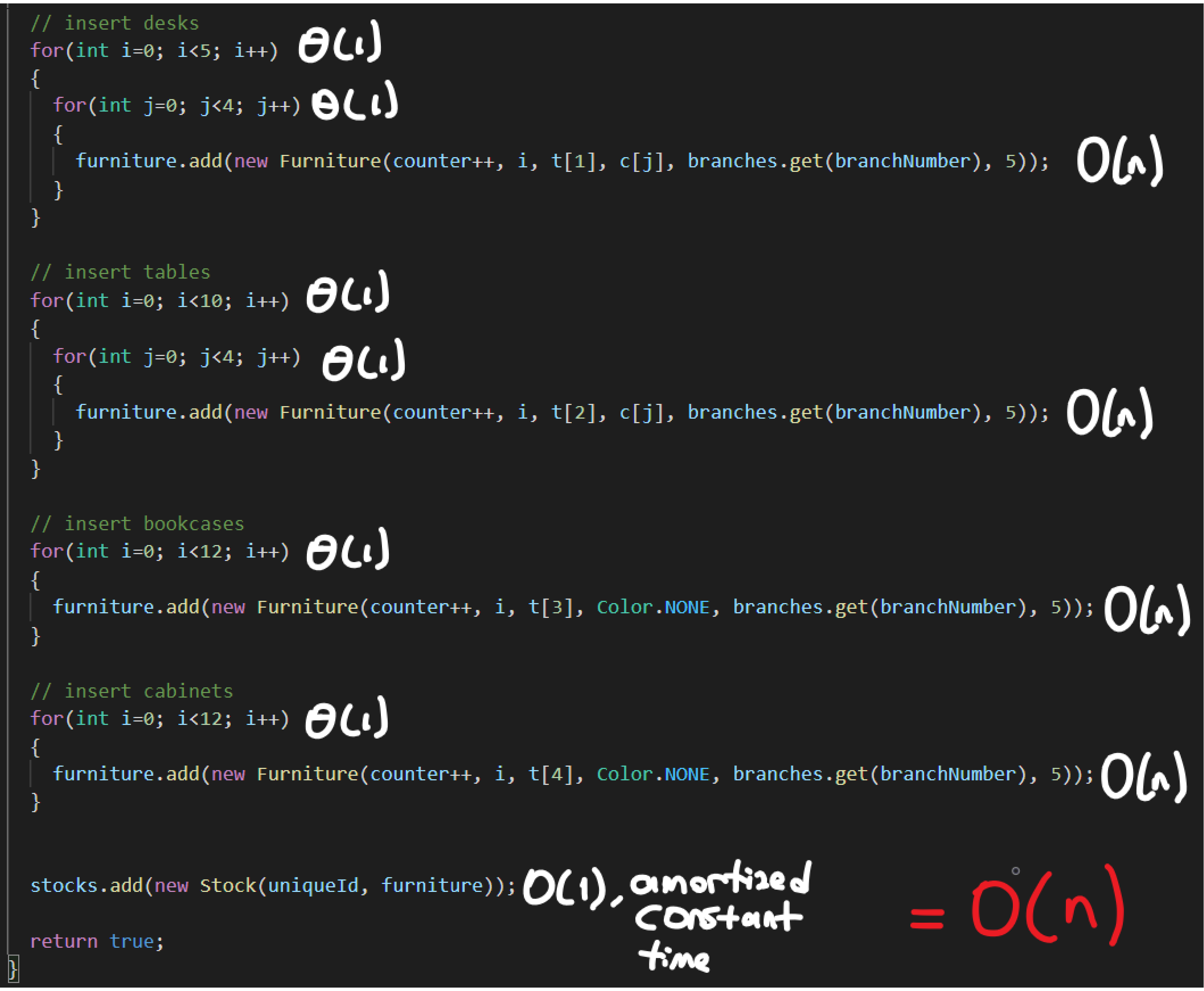
```
ProductId : 21
Amount : 4
Home address : adress
Phone number : 053543
```

Part 2 :

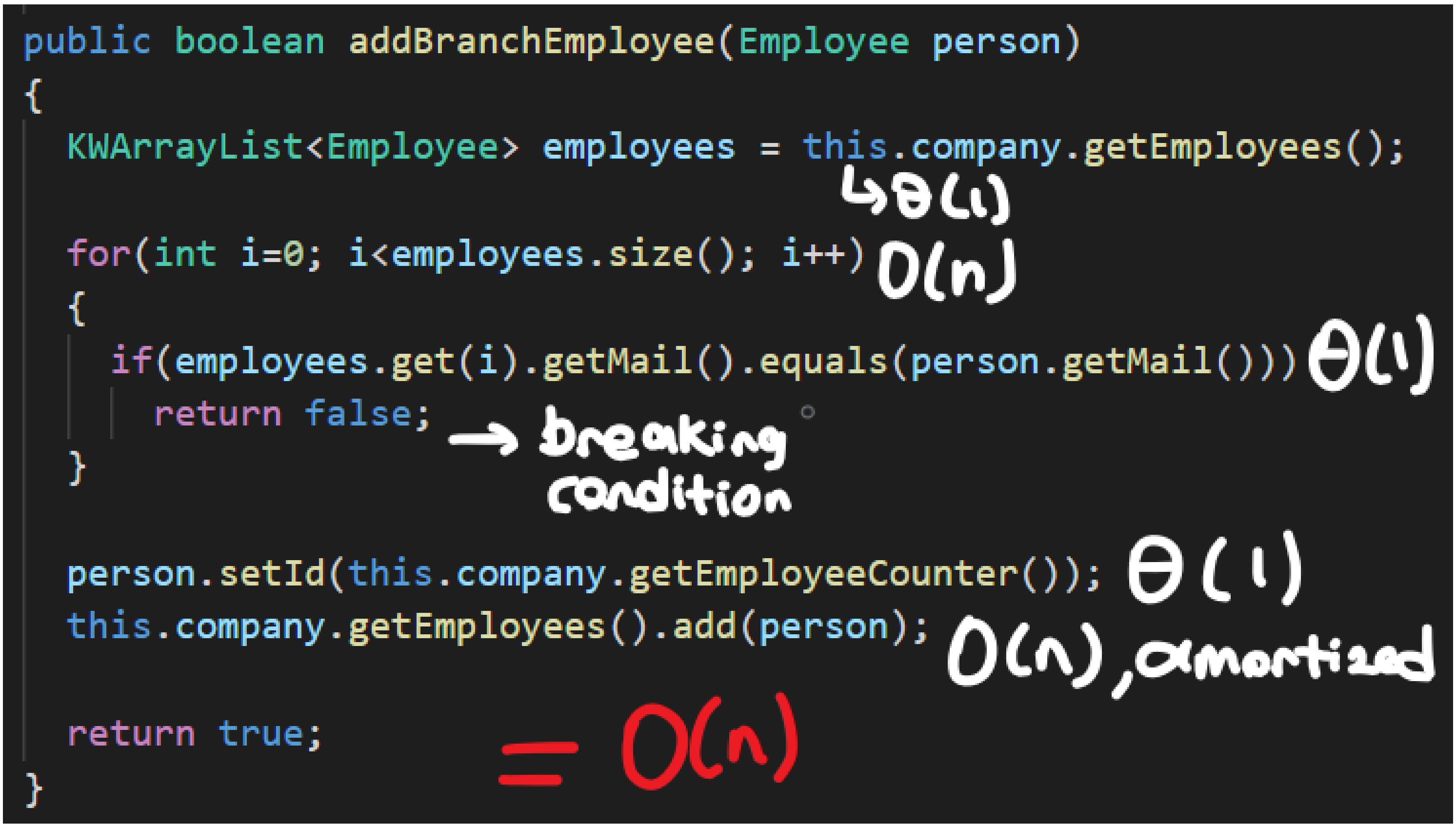
I did not implement the constant time getters and setters



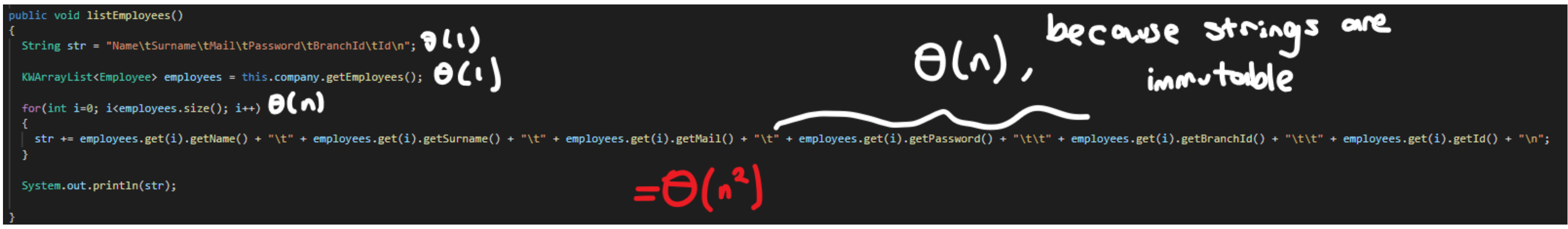
```java
public boolean addBranch()
{
  KWLinkedList<Branch> branches = this.company.getBranches();
  KWArrayList<Stock> stocks = this.company.getStocks();
  int branchNumber = branches.size();
  int uniqueId = this.company.getBranchCounter();

  branches.add(new Branch(uniqueId, uniqueId));

  Type t[] = Type.values();
  Color c[] = Color.values();
  HybridList<Furniture> furniture = new HybridList<Furniture>();

  int counter = 0;

  // insert chairs
  for(int i=0; i<7; i++)
  {
    for(int j=0; j<5; j++)
    {
      furniture.add(new Furniture(counter++, i, t[0], c[j], branches.get(branchNumber), 5));
    }
  }
```



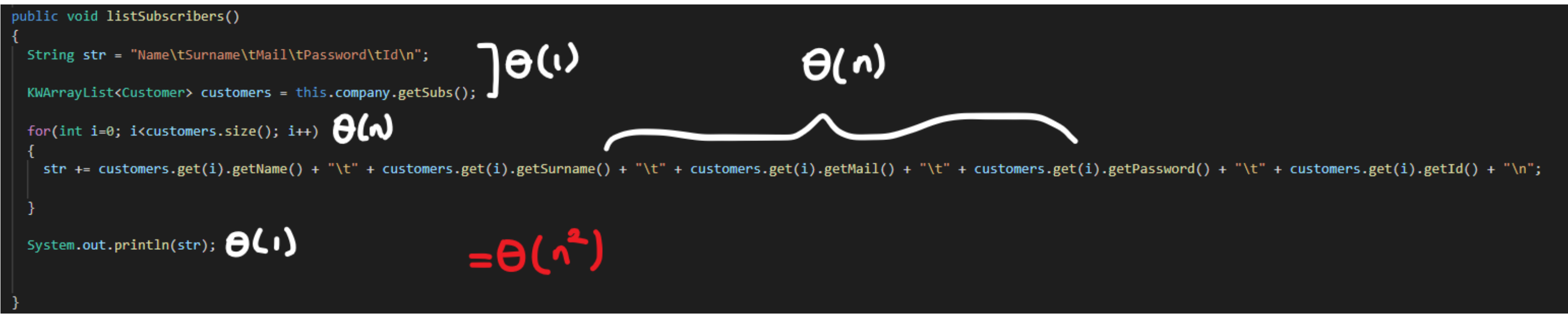
```
// insert desks
for(int i=0; i<5; i++) θ(1)
{
  for(int j=0; j<4; j++) θ(1)
  {
    furniture.add(new Furniture(counter++, i, t[1], c[j], branches.get(branchNumber), 5)); O(n)
  }
}

// insert tables
for(int i=0; i<10; i++) θ(1)
{
  for(int j=0; j<4; j++) θ(1)
  {
    furniture.add(new Furniture(counter++, i, t[2], c[j], branches.get(branchNumber), 5)); O(n)
  }
}

// insert bookcases
for(int i=0; i<12; i++) θ(1)
{
  furniture.add(new Furniture(counter++, i, t[3], Color.NONE, branches.get(branchNumber), 5)); O(n)
}

// insert cabinets
for(int i=0; i<12; i++) θ(1)
{
  furniture.add(new Furniture(counter++, i, t[4], Color.NONE, branches.get(branchNumber), 5)); O(n)
}


stocks.add(new Stock(uniqueId, furniture)); O(1), amortized
                                              constant
                                              time          = O(n)

return true;
}
```



```
public boolean addBranchEmployee(Employee person)
{
    KWArrayList<Employee> employees = this.company.getEmployees();
                                      ↳ Θ(1)
    for(int i=0; i<employees.size(); i++) O(n)
    {
        if(employees.get(i).getMail().equals(person.getMail())) Θ(1)
            return false; → breaking condition
    }

    person.setId(this.company.getEmployeeCounter()); Θ(1)
    this.company.getEmployees().add(person); O(n), amortized

    return true;          = O(n)
}
```

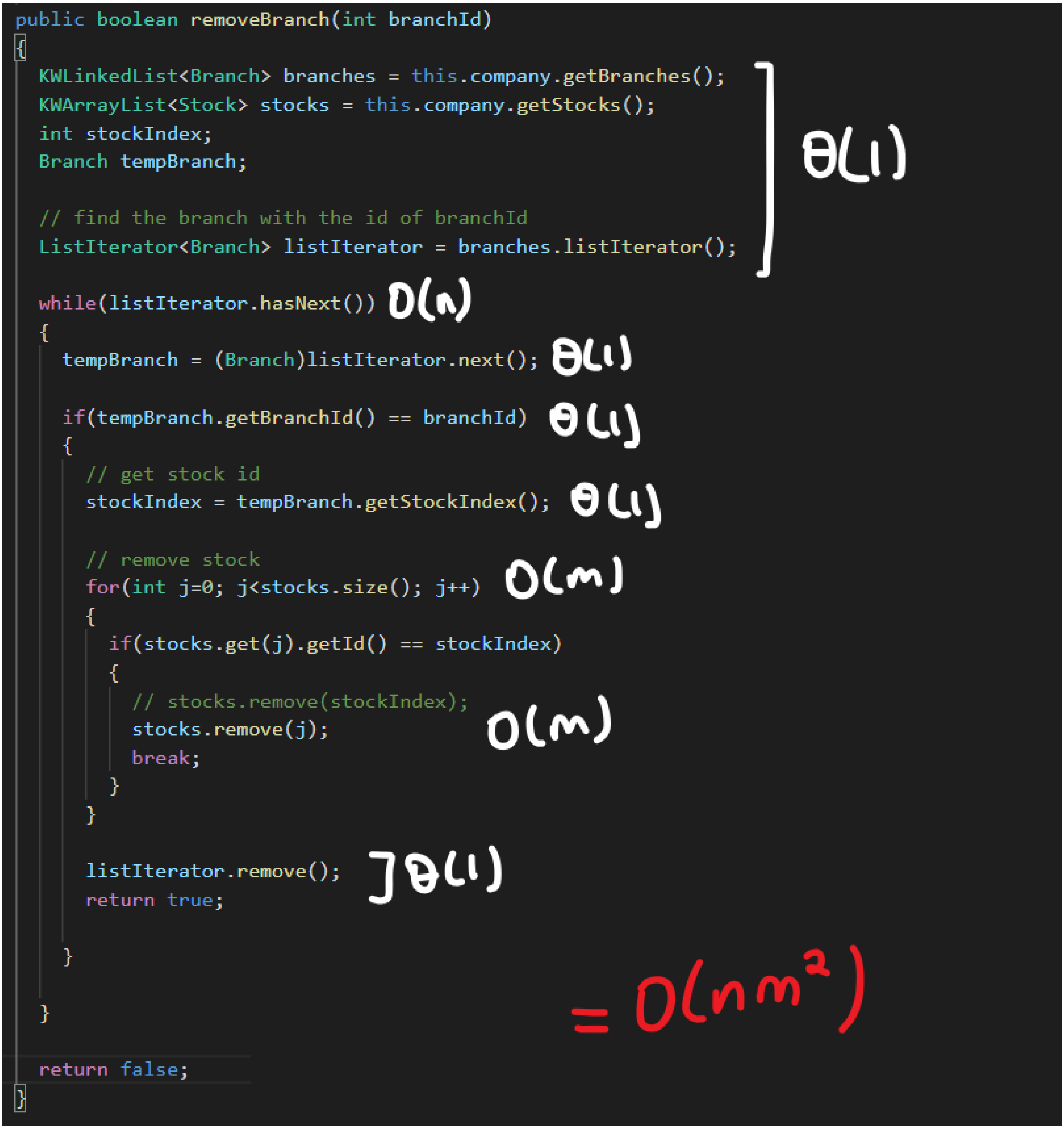


```java
public void listEmployees()
{
  String str = "Name\tSurname\tMail\tPassword\tBranchId\tId\n";

  KWArrayList<Employee> employees = this.company.getEmployees();

  for(int i=0; i<employees.size(); i++)
  {
    str += employees.get(i).getName() + "\t" + employees.get(i).getSurname() + "\t" + employees.get(i).getMail() + "\t" + employees.get(i).getPassword() + "\t\t" + employees.get(i).getBranchId() + "\t\t" + employees.get(i).getId() + "\n";
  }

  System.out.println(str);

}
```

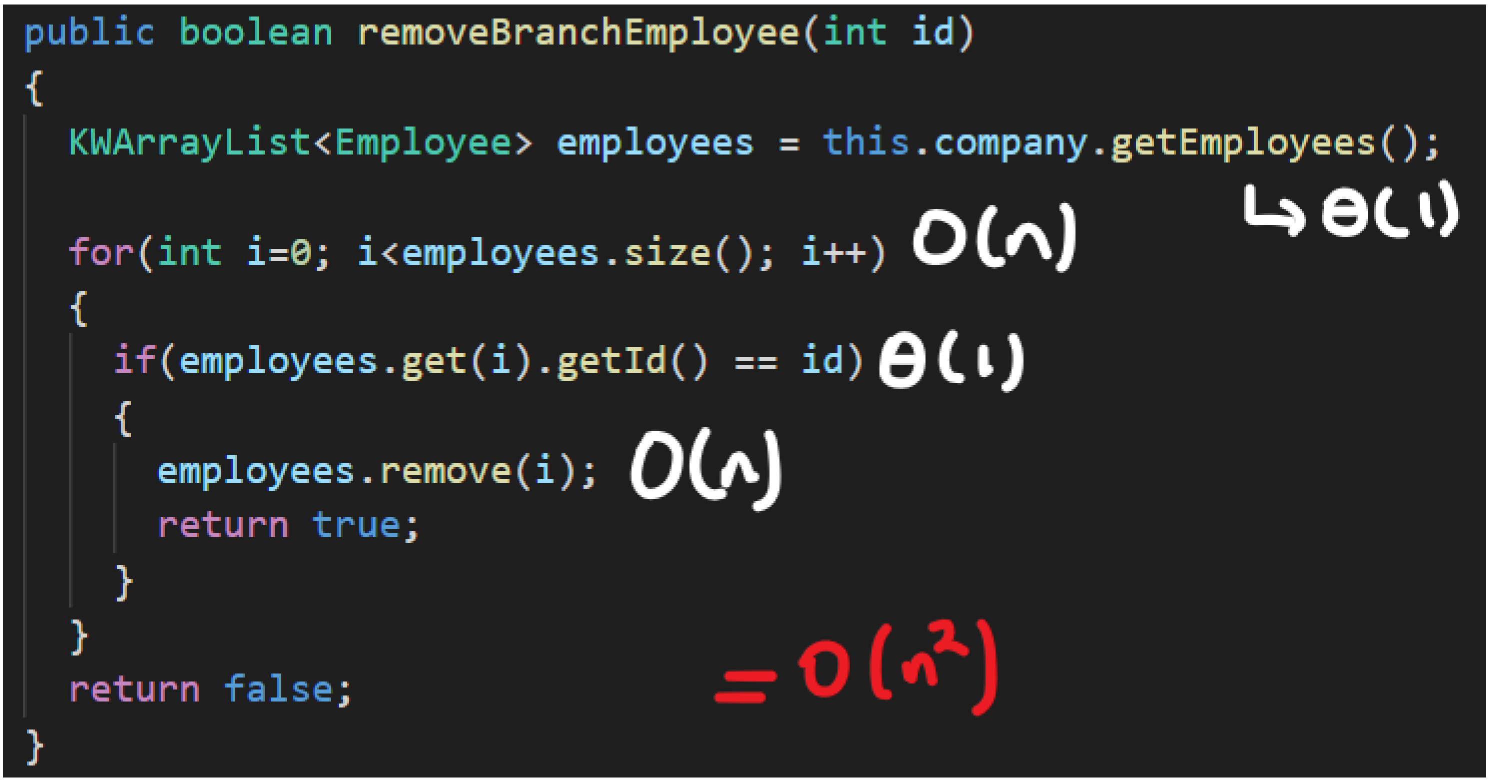


```java
public void listSubscribers()
{
  String str = "Name\tSurname\tMail\tPassword\tId\n";

  KWArrayList<Customer> customers = this.company.getSubs();

  for(int i=0; i<customers.size(); i++)
  {
    str += customers.get(i).getName() + "\t" + customers.get(i).getSurname() + "\t" + customers.get(i).getMail() + "\t" + customers.get(i).getPassword() + "\t" + customers.get(i).getId() + "\n";

  }

  System.out.println(str);


}
```

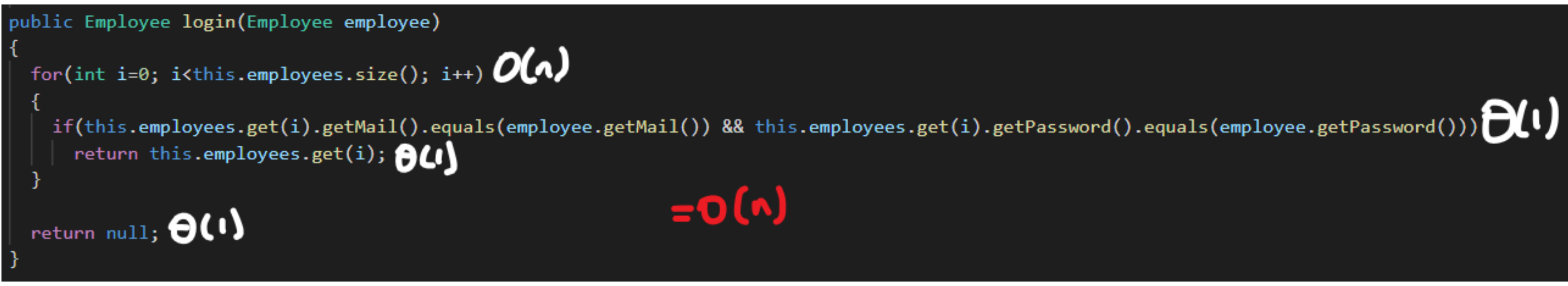
n is the number of branches and m is the size of the individual stocks



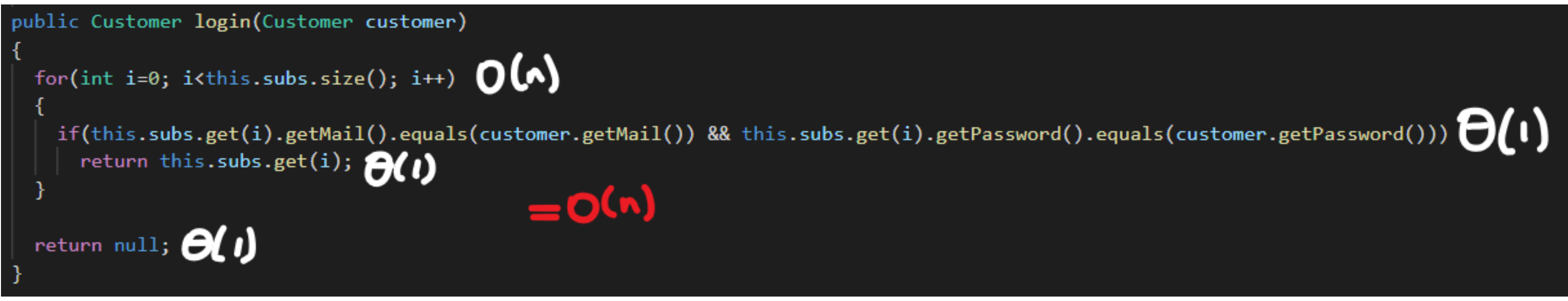
```
public boolean removeBranch(int branchId)
{
    KWLinkedList<Branch> branches = this.company.getBranches();
    KWArrayList<Stock> stocks = this.company.getStocks();
    int stockIndex;
    Branch tempBranch;

    // find the branch with the id of branchId
    ListIterator<Branch> listIterator = branches.listIterator();

    while(listIterator.hasNext())
    {
        tempBranch = (Branch)listIterator.next();

        if(tempBranch.getBranchId() == branchId)
        {
            // get stock id
            stockIndex = tempBranch.getStockIndex();

            // remove stock
            for(int j=0; j<stocks.size(); j++)
            {
                if(stocks.get(j).getId() == stockIndex)
                {
                    // stocks.remove(stockIndex);
                    stocks.remove(j);
                    break;
                }
            }

            listIterator.remove();
            return true;

        }

    }

    return false;
}
```



```
public boolean removeBranchEmployee(int id)
{
  KWArrayList<Employee> employees = this.company.getEmployees();
                                                          ↳ Θ(1)
  for(int i=0; i<employees.size(); i++) O(n)
  {
    if(employees.get(i).getId() == id) Θ(1)
    {
      employees.remove(i); O(n)
      return true;
    }
  }
  return false;                     = O(n²)
}
```

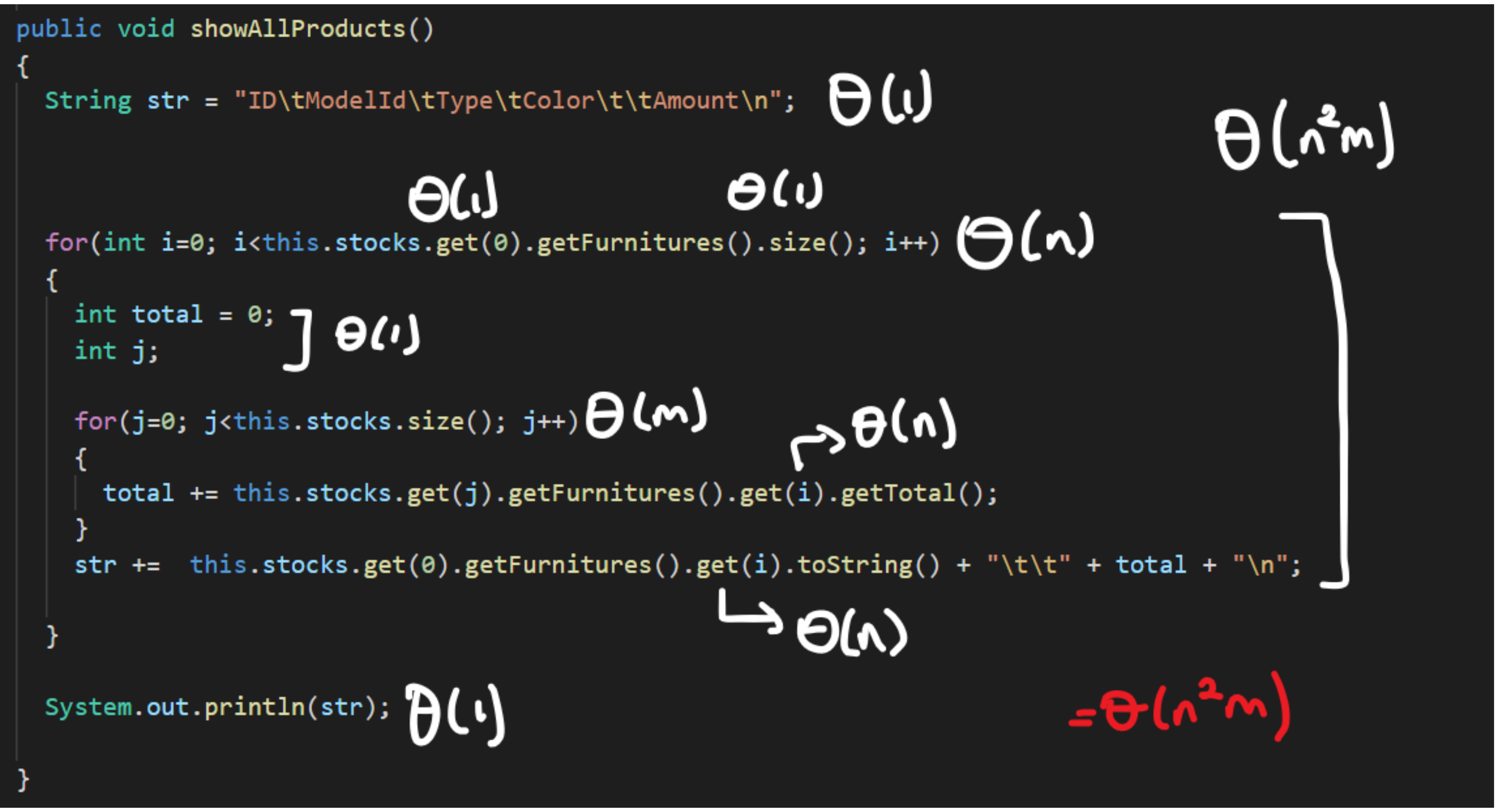


```
public Employee login(Employee employee)
{
  for(int i=0; i<this.employees.size(); i++) O(n)
  {
    if(this.employees.get(i).getMail().equals(employee.getMail()) && this.employees.get(i).getPassword().equals(employee.getPassword())) Θ(1)
      return this.employees.get(i); Θ(1)
  }
                                                    =O(n)
  return null; Θ(1)
}
```

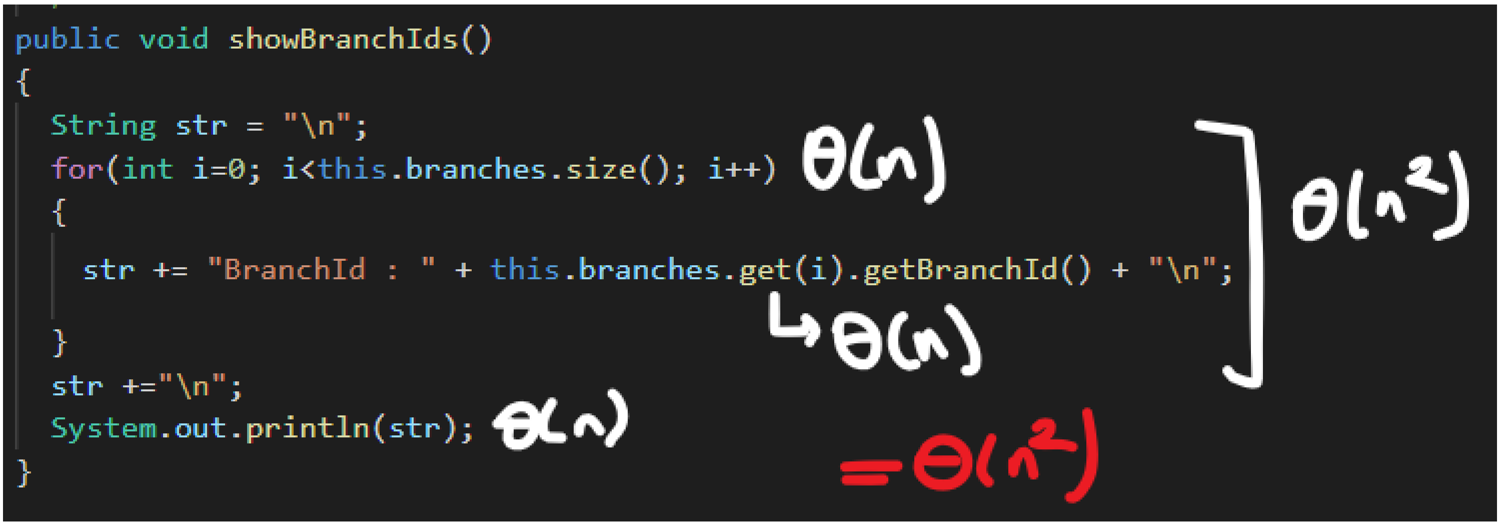


```
public Customer login(Customer customer)
{
  for(int i=0; i<this.subs.size(); i++) O(n)
  {
    if(this.subs.get(i).getMail().equals(customer.getMail()) && this.subs.get(i).getPassword().equals(customer.getPassword())) Θ(1)
      return this.subs.get(i); Θ(1)
  }
                                        =O(n)
  return null; Θ(1)
}
```

n is the number of furnitures and m is the number of stocks

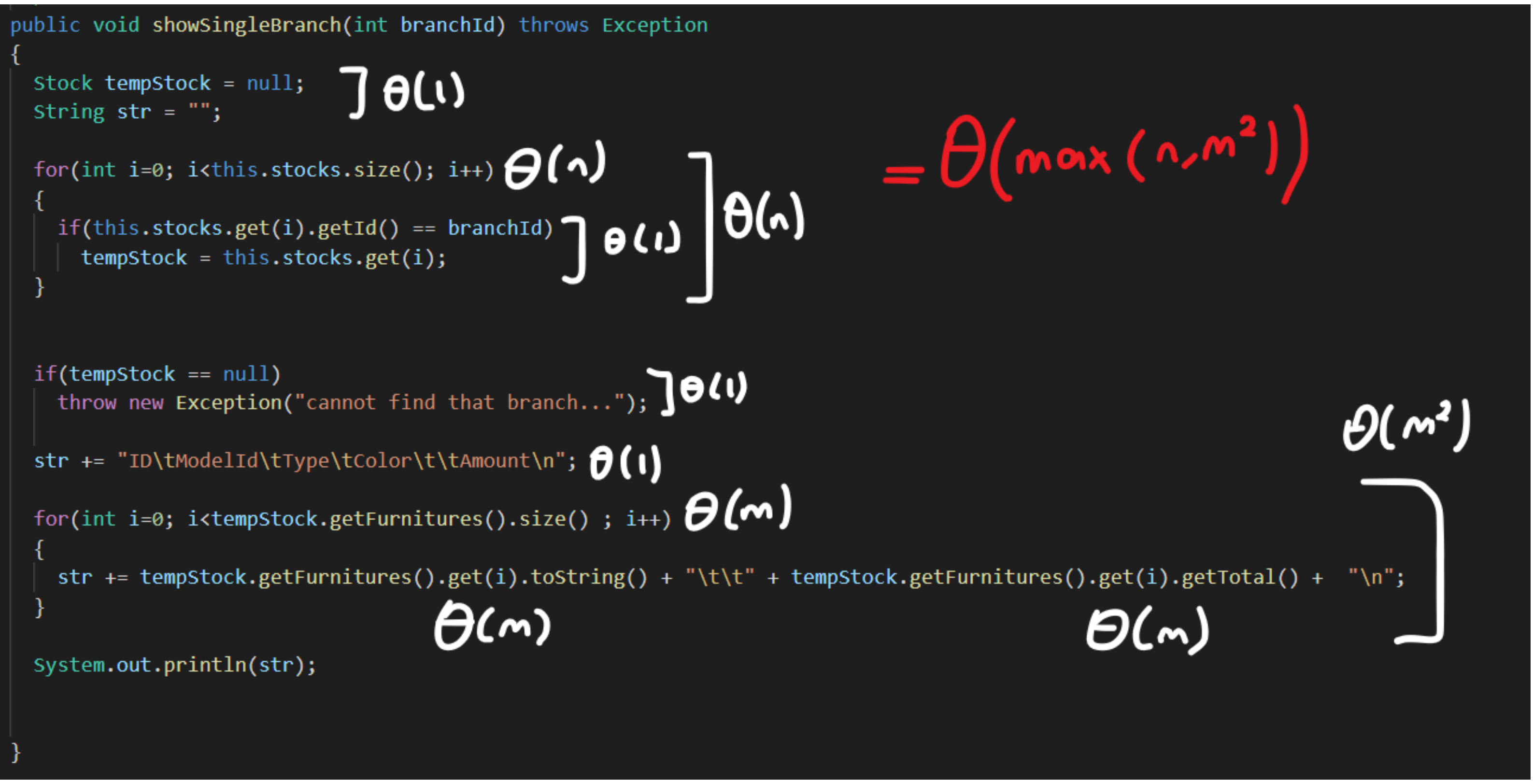


```
public void showAllProducts()
{
  String str = "ID\tModelId\tType\tColor\t\tAmount\n";


  for(int i=0; i<this.stocks.get(0).getFurnitures().size(); i++)
  {
    int total = 0;
    int j;

    for(j=0; j<this.stocks.size(); j++)
    {
      total += this.stocks.get(j).getFurnitures().get(i).getTotal();
    }
    str +=  this.stocks.get(0).getFurnitures().get(i).toString() + "\t\t" + total + "\n";

  }

  System.out.println(str);
}
```

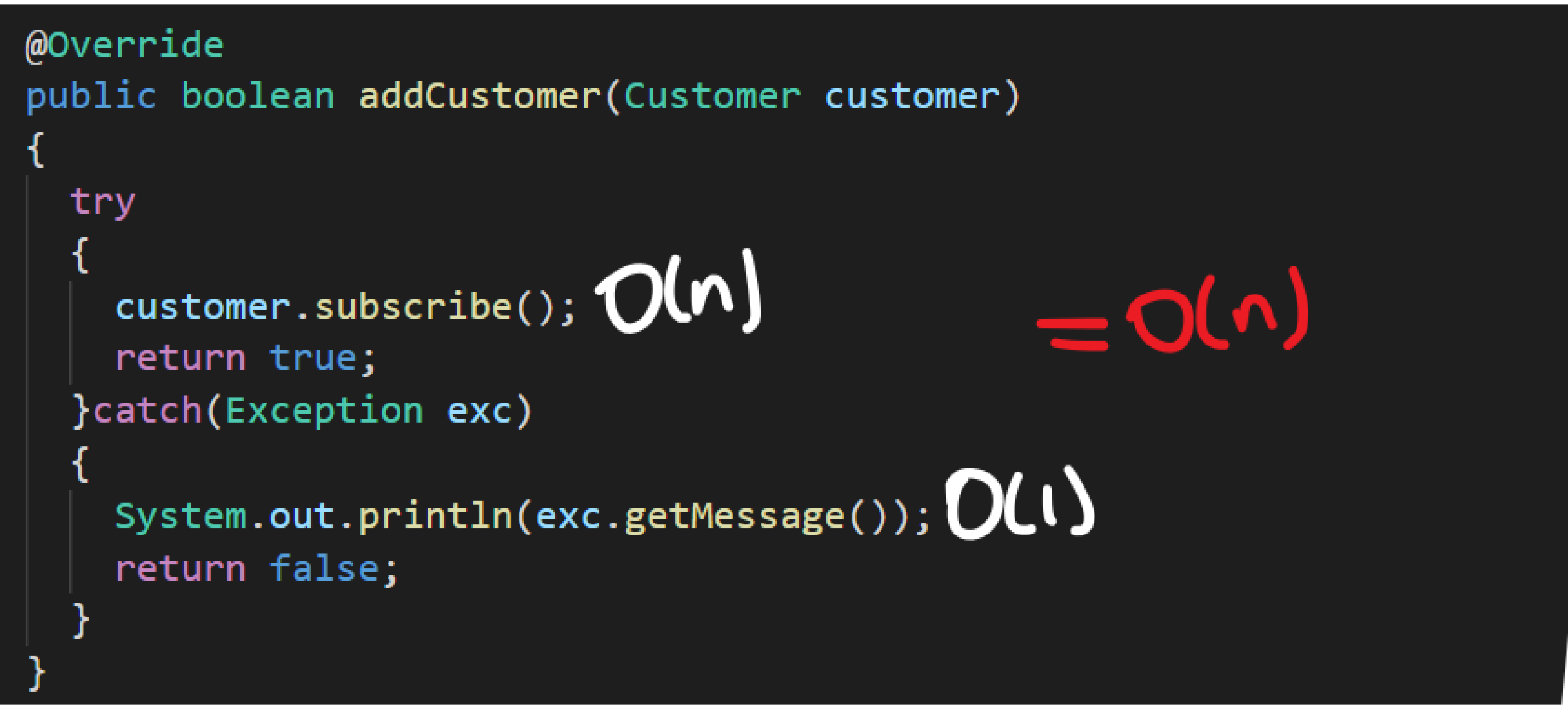


```
public void showBranchIds()
{
  String str = "\n";
  for(int i=0; i<this.branches.size(); i++)
  {
    str += "BranchId : " + this.branches.get(i).getBranchId() + "\n";

  }
  str +="\n";
  System.out.println(str);
}
```

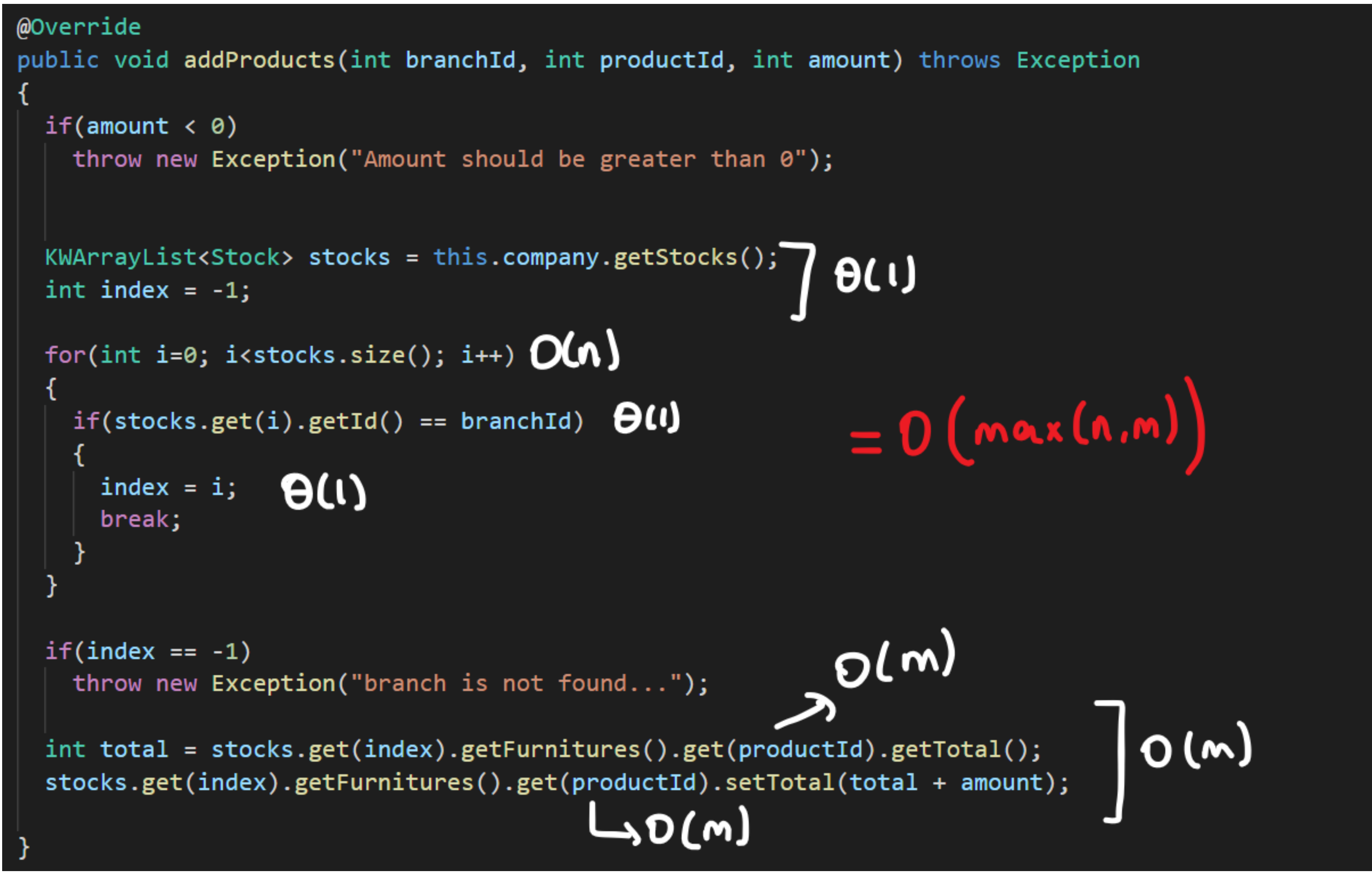
n is the number of stocks, m is the size of the furnitures



```
public void showSingleBranch(int branchId) throws Exception
{
  Stock tempStock = null;
  String str = "";

  for(int i=0; i<this.stocks.size(); i++)
  {
    if(this.stocks.get(i).getId() == branchId)
      tempStock = this.stocks.get(i);
  }


  if(tempStock == null)
    throw new Exception("cannot find that branch...");

  str += "ID\tModelId\tType\tColor\t\tAmount\n";

  for(int i=0; i<tempStock.getFurnitures().size() ; i++)
  {
    str += tempStock.getFurnitures().get(i).toString() + "\t\t" + tempStock.getFurnitures().get(i).getTotal() +  "\n";
  }

  System.out.println(str);


}
```



```
@Override
public boolean addCustomer(Customer customer)
{
  try
  {
    customer.subscribe();
    return true;
  }catch(Exception exc)
  {
    System.out.println(exc.getMessage());
    return false;
  }
}
```



```java
@Override
public void addProducts(int branchId, int productId, int amount) throws Exception
{
  if(amount < 0)
    throw new Exception("Amount should be greater than 0");


  KWArrayList<Stock> stocks = this.company.getStocks();
  int index = -1;

  for(int i=0; i<stocks.size(); i++)
  {
    if(stocks.get(i).getId() == branchId)
    {
      index = i;
      break;
    }
  }

  if(index == -1)
    throw new Exception("branch is not found...");

  int total = stocks.get(index).getFurnitures().get(productId).getTotal();
  stocks.get(index).getFurnitures().get(productId).setTotal(total + amount);
}
```

$\Theta(1)$

$O(n)$

$\Theta(1)$

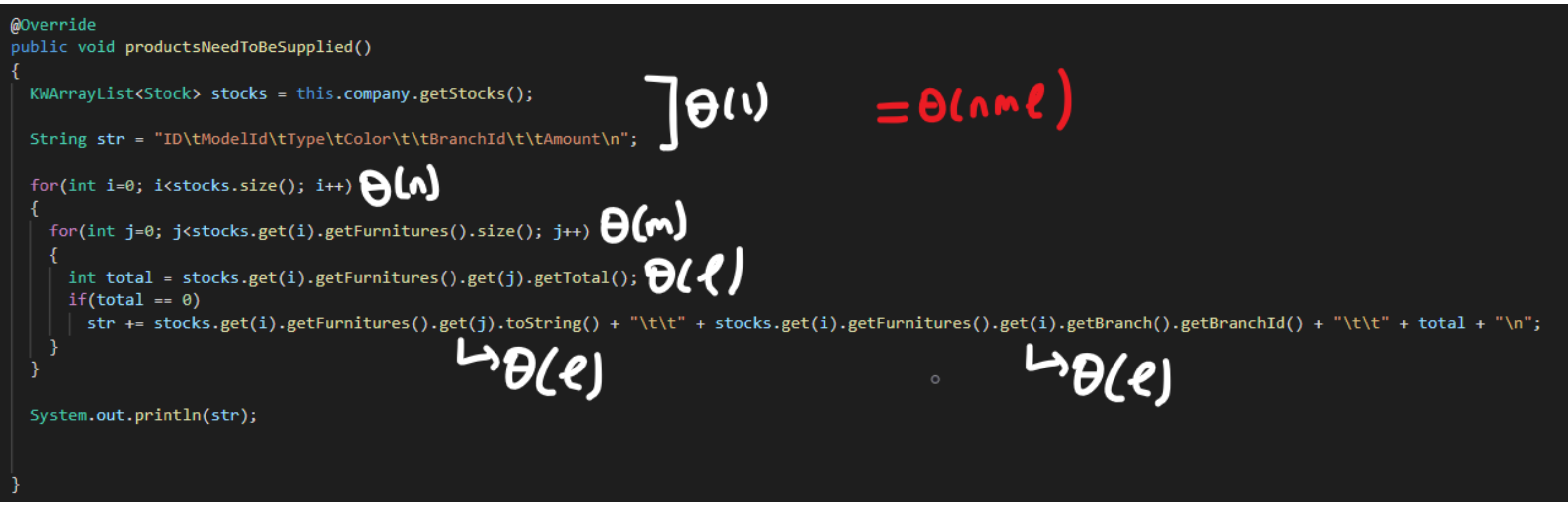
$\Theta(1)$

$= O(\max(n,m))$

$O(m)$

$O(m)$

$\hookrightarrow O(m)$



```java
@Override
public void productsNeedToBeSupplied()
{
  KWArrayList<Stock> stocks = this.company.getStocks();

  String str = "ID\tModelId\tType\tColor\t\tBranchId\t\tAmount\n";

  for(int i=0; i<stocks.size(); i++)
  {
    for(int j=0; j<stocks.get(i).getFurnitures().size(); j++)
    {
      int total = stocks.get(i).getFurnitures().get(j).getTotal();
      if(total == 0)
        str += stocks.get(i).getFurnitures().get(j).toString() + "\t\t" + stocks.get(i).getFurnitures().get(i).getBranch().getBranchId() + "\t\t" + total + "\n";
    }
  }

  System.out.println(str);


}
```

$\Theta(1)$

$= \Theta(nml)$

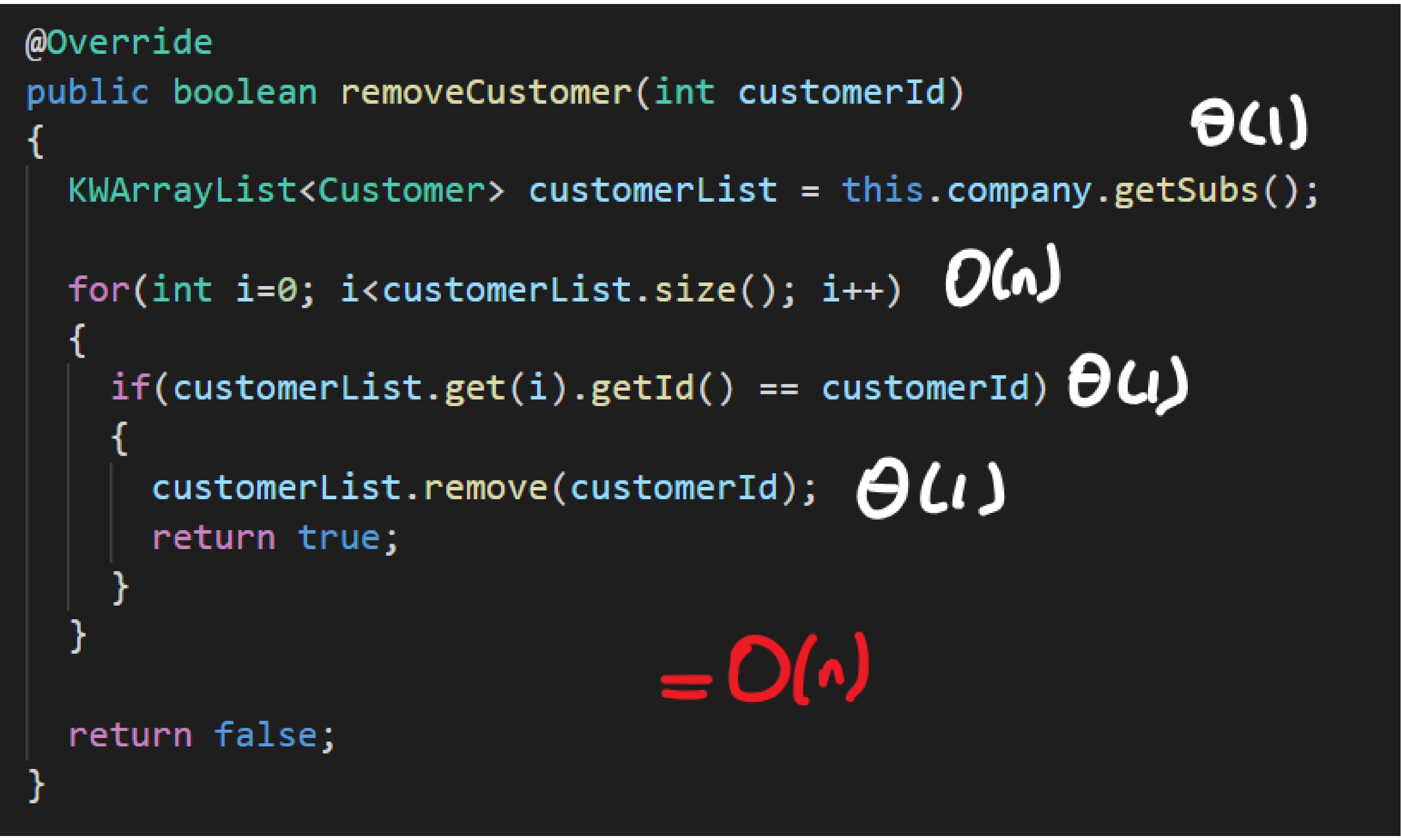
$\Theta(n)$

$\Theta(m)$

$\Theta(l)$

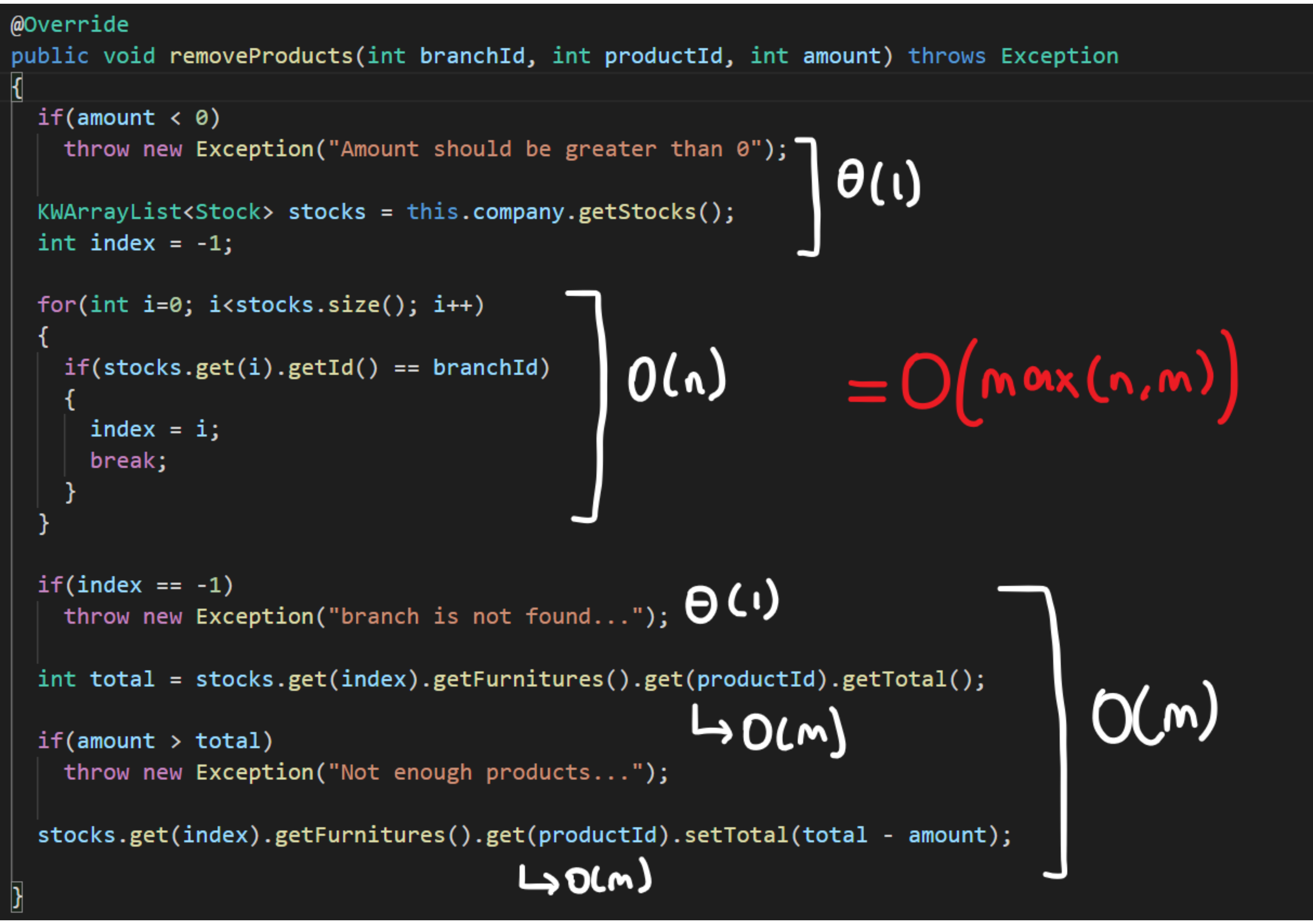
$\hookrightarrow \Theta(l)$

$\hookrightarrow \Theta(l)$

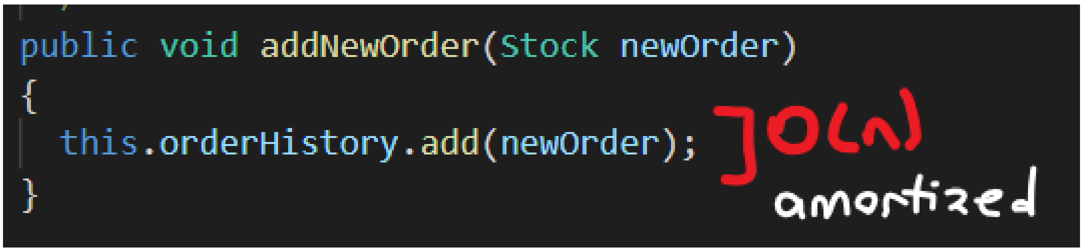


```
@Override
public boolean removeCustomer(int customerId)
{                                                         Θ(1)
  KWArrayList<Customer> customerList = this.company.getSubs();

  for(int i=0; i<customerList.size(); i++)  O(n)
  {
    if(customerList.get(i).getId() == customerId) Θ(1)
    {
      customerList.remove(customerId);  Θ(1)
      return true;
    }
  }
                                =O(n)
  return false;
}
```

n is the number of stocks, m is the number of furnitures in a stock

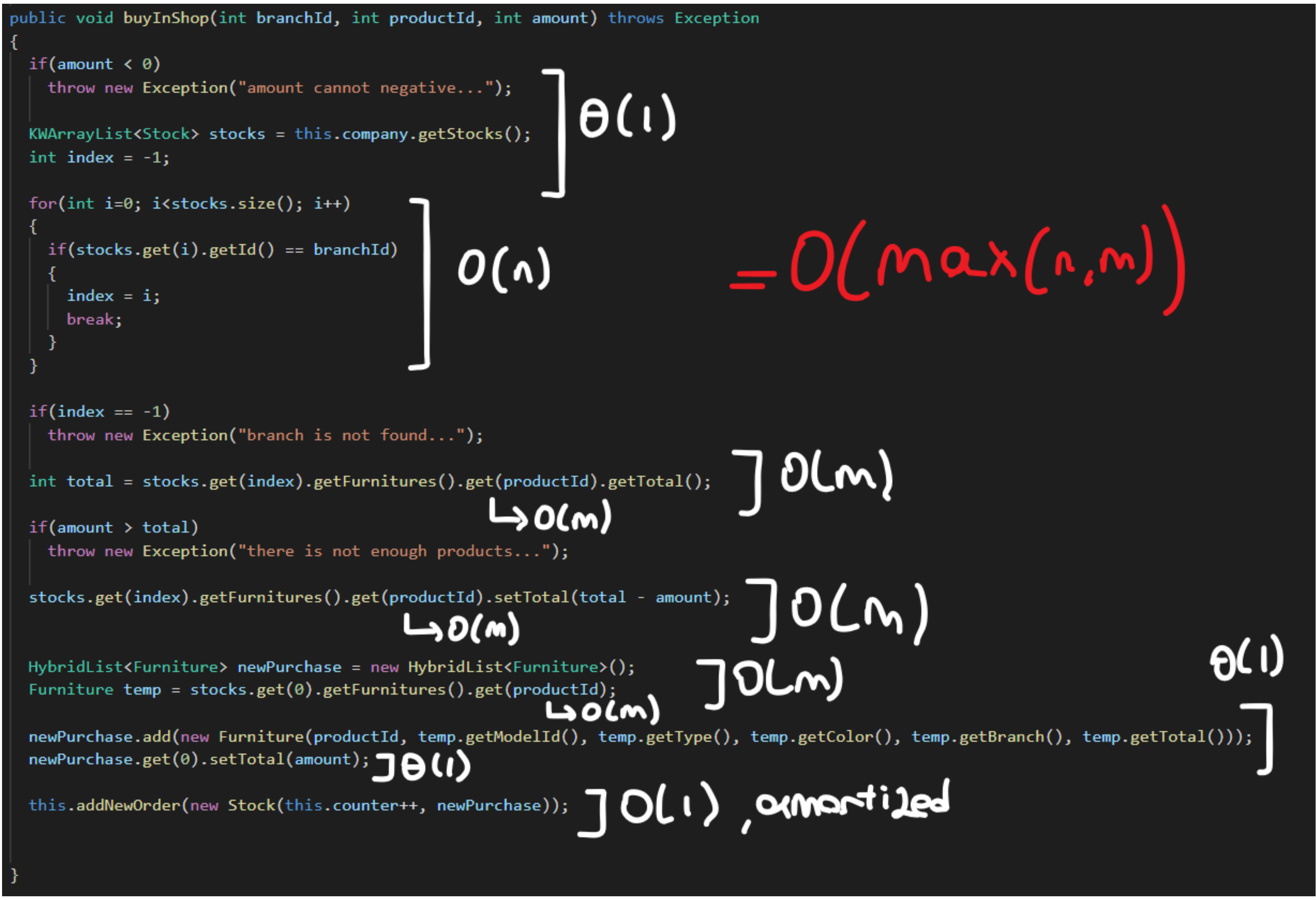


```
@Override
public void removeProducts(int branchId, int productId, int amount) throws Exception
{
    if(amount < 0)
        throw new Exception("Amount should be greater than 0");

    KWArrayList<Stock> stocks = this.company.getStocks();
    int index = -1;

    for(int i=0; i<stocks.size(); i++)
    {
        if(stocks.get(i).getId() == branchId)
        {
            index = i;
            break;
        }
    }

    if(index == -1)
        throw new Exception("branch is not found...");

    int total = stocks.get(index).getFurnitures().get(productId).getTotal();

    if(amount > total)
        throw new Exception("Not enough products...");

    stocks.get(index).getFurnitures().get(productId).setTotal(total - amount);
}
```



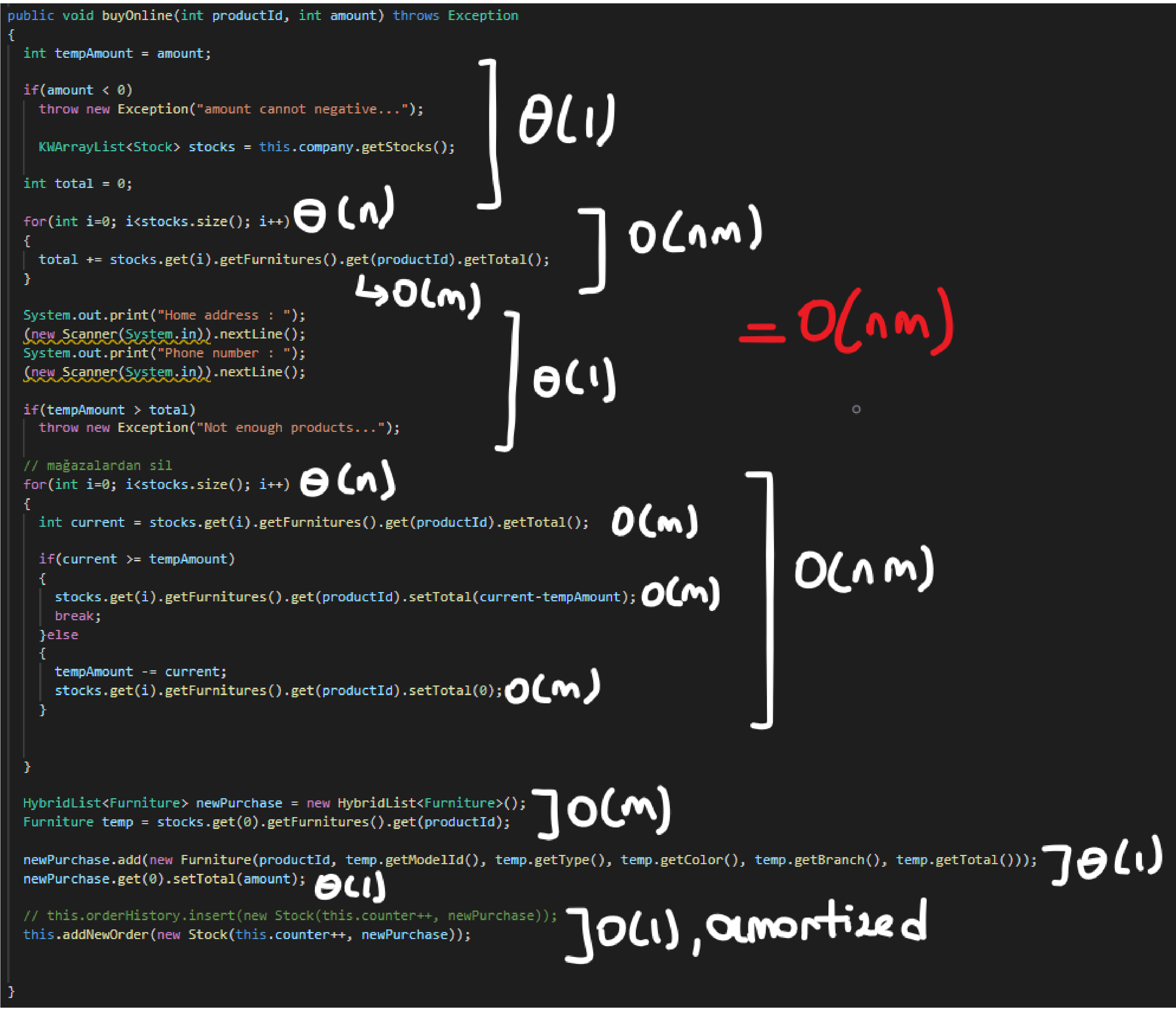
```
public void addNewOrder(Stock newOrder)
{
    this.orderHistory.add(newOrder);
}
```

n is the number of stocks, m is the number of furnitures in a stock



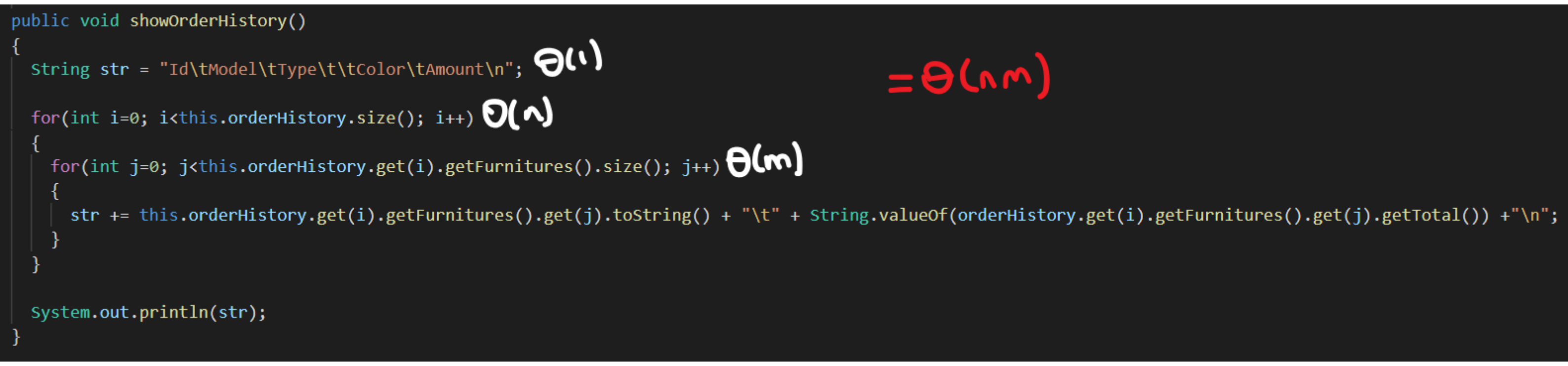
```
public void buyInShop(int branchId, int productId, int amount) throws Exception
{
  if(amount < 0)
    throw new Exception("amount cannot negative...");

  KWArrayList<Stock> stocks = this.company.getStocks();
  int index = -1;

  for(int i=0; i<stocks.size(); i++)
  {
    if(stocks.get(i).getId() == branchId)
    {
      index = i;
      break;
    }
  }

  if(index == -1)
    throw new Exception("branch is not found...");

  int total = stocks.get(index).getFurnitures().get(productId).getTotal();

  if(amount > total)
    throw new Exception("there is not enough products...");

  stocks.get(index).getFurnitures().get(productId).setTotal(total - amount);

  HybridList<Furniture> newPurchase = new HybridList<Furniture>();
  Furniture temp = stocks.get(0).getFurnitures().get(productId);

  newPurchase.add(new Furniture(productId, temp.getModelId(), temp.getType(), temp.getColor(), temp.getBranch(), temp.getTotal()));
  newPurchase.get(0).setTotal(amount);

  this.addNewOrder(new Stock(this.counter++, newPurchase));

}
```

n is the number of stocks, m is the number of furnitures in a stock

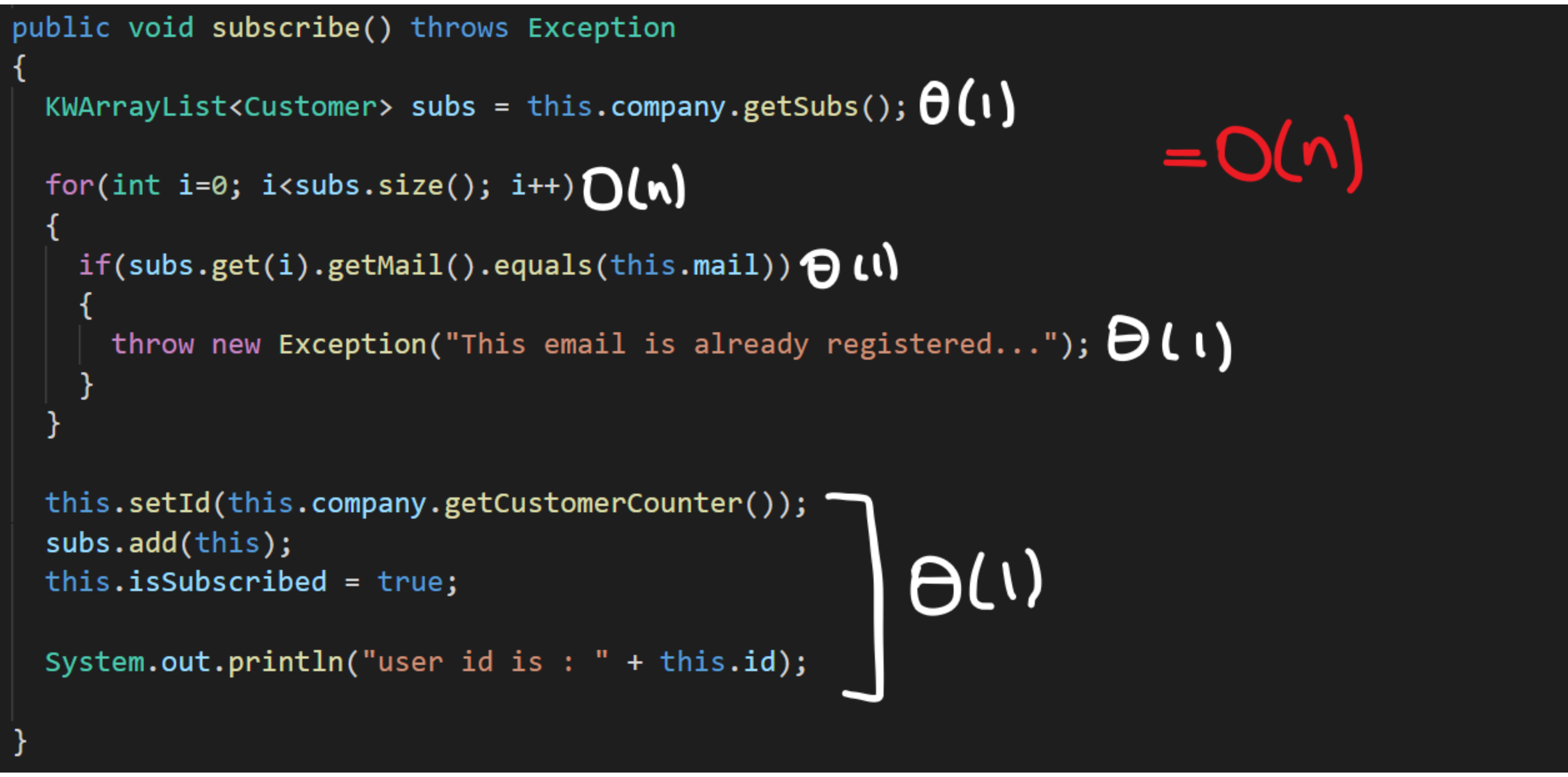


```java
public void buyOnline(int productId, int amount) throws Exception
{
  int tempAmount = amount;

  if(amount < 0)
    throw new Exception("amount cannot negative...");

    KWArrayList<Stock> stocks = this.company.getStocks();

  int total = 0;

  for(int i=0; i<stocks.size(); i++)
  {
    total += stocks.get(i).getFurnitures().get(productId).getTotal();
  }

  System.out.print("Home address : ");
  (new Scanner(System.in)).nextLine();
  System.out.print("Phone number : ");
  (new Scanner(System.in)).nextLine();

  if(tempAmount > total)
    throw new Exception("Not enough products...");

  // mağazalardan sil
  for(int i=0; i<stocks.size(); i++)
  {
    int current = stocks.get(i).getFurnitures().get(productId).getTotal();

    if(current >= tempAmount)
    {
      stocks.get(i).getFurnitures().get(productId).setTotal(current-tempAmount);
      break;
    }else
    {
      tempAmount -= current;
      stocks.get(i).getFurnitures().get(productId).setTotal(0);
    }


  }

  HybridList<Furniture> newPurchase = new HybridList<Furniture>();
  Furniture temp = stocks.get(0).getFurnitures().get(productId);

  newPurchase.add(new Furniture(productId, temp.getModelId(), temp.getType(), temp.getColor(), temp.getBranch(), temp.getTotal()));
  newPurchase.get(0).setTotal(amount);

  // this.orderHistory.insert(new Stock(this.counter++, newPurchase));
  this.addNewOrder(new Stock(this.counter++, newPurchase));

}
```

n is the number of stocks, m is the number of furnitures in a stock



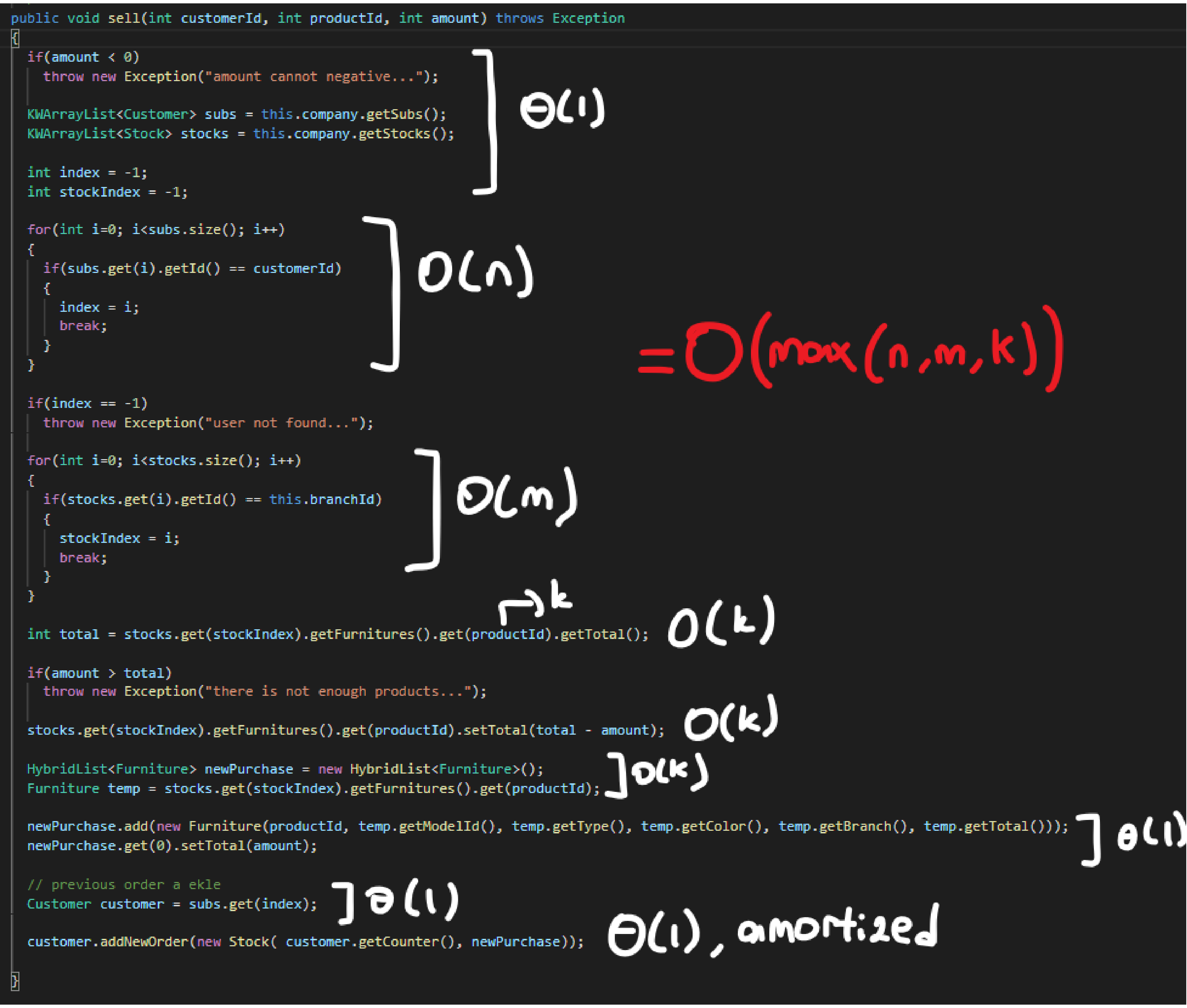
```java
public void showOrderHistory()
{
  String str = "Id\tModel\tType\t\tColor\tAmount\n";

  for(int i=0; i<this.orderHistory.size(); i++)
  {
    for(int j=0; j<this.orderHistory.get(i).getFurnitures().size(); j++)
    {
      str += this.orderHistory.get(i).getFurnitures().get(j).toString() + "\t" + String.valueOf(orderHistory.get(i).getFurnitures().get(j).getTotal()) +"\n";
    }
  }

  System.out.println(str);
}
```



```
public void subscribe() throws Exception
{
  KWArrayList<Customer> subs = this.company.getSubs(); θ(1)

  for(int i=0; i<subs.size(); i++) O(n)
  {
    if(subs.get(i).getMail().equals(this.mail)) θ(1)
    {
      throw new Exception("This email is already registered..."); θ(1)
    }
  }

  this.setId(this.company.getCustomerCounter());
  subs.add(this);
  this.isSubscribed = true;                            θ(1)

  System.out.println("user id is : " + this.id);

}
```

=O(n)

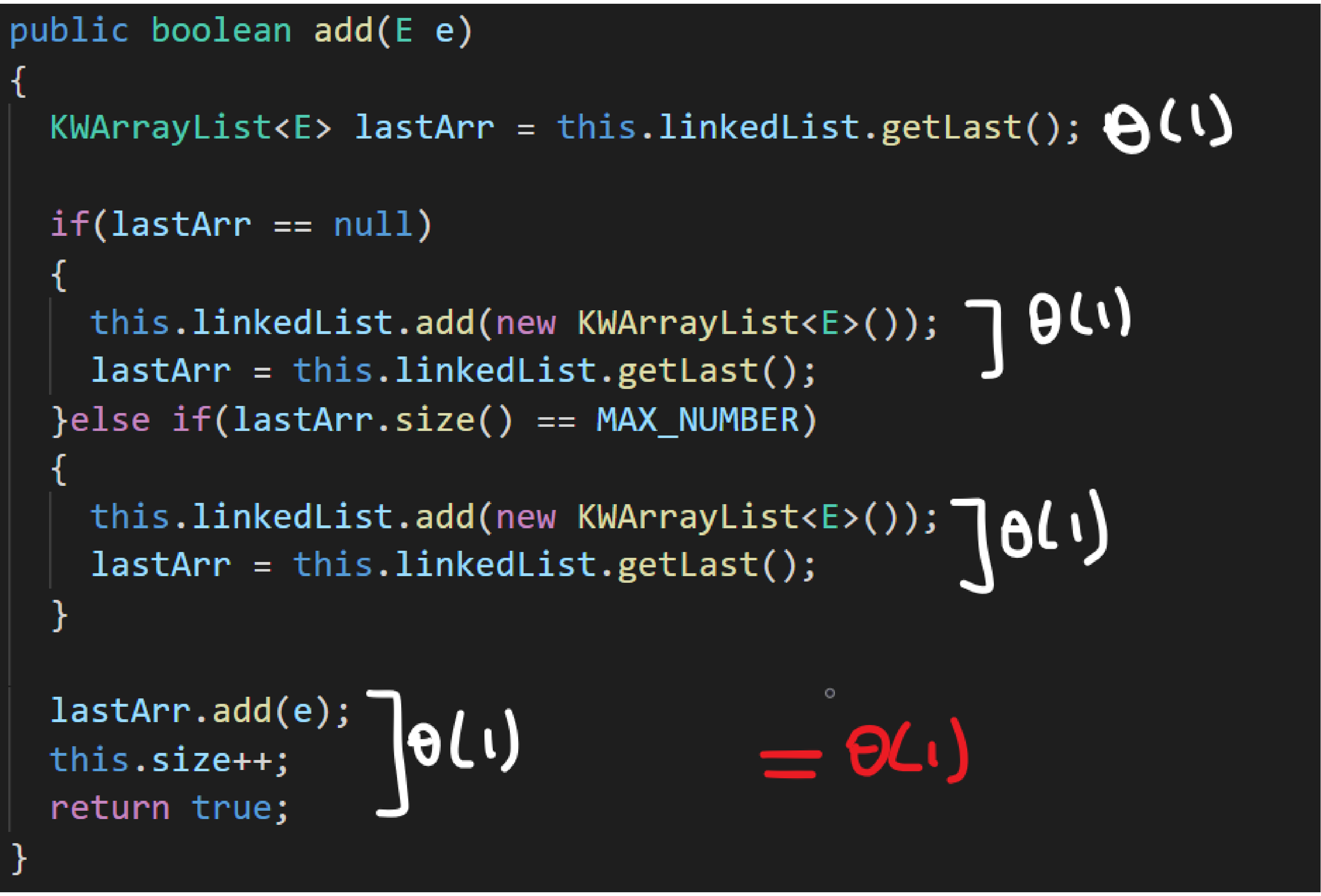
n is the number of subscribers(customers),
m is the size of the stocks
k is the number of furnitures in a stock



```
public void sell(int customerId, int productId, int amount) throws Exception
{
    if(amount < 0)
        throw new Exception("amount cannot negative...");

    KWArrayList<Customer> subs = this.company.getSubs();
    KWArrayList<Stock> stocks = this.company.getStocks();

    int index = -1;
    int stockIndex = -1;

    for(int i=0; i<subs.size(); i++)
    {
        if(subs.get(i).getId() == customerId)
        {
            index = i;
            break;
        }
    }

    if(index == -1)
        throw new Exception("user not found...");

    for(int i=0; i<stocks.size(); i++)
    {
        if(stocks.get(i).getId() == this.branchId)
        {
            stockIndex = i;
            break;
        }
    }

    int total = stocks.get(stockIndex).getFurnitures().get(productId).getTotal();

    if(amount > total)
        throw new Exception("there is not enough products...");

    stocks.get(stockIndex).getFurnitures().get(productId).setTotal(total - amount);

    HybridList<Furniture> newPurchase = new HybridList<Furniture>();
    Furniture temp = stocks.get(stockIndex).getFurnitures().get(productId);

    newPurchase.add(new Furniture(productId, temp.getModelId(), temp.getType(), temp.getColor(), temp.getBranch(), temp.getTotal()));
    newPurchase.get(0).setTotal(amount);

    // previous order a ekle
    Customer customer = subs.get(index);

    customer.addNewOrder(new Stock( customer.getCounter(), newPurchase));

}
```

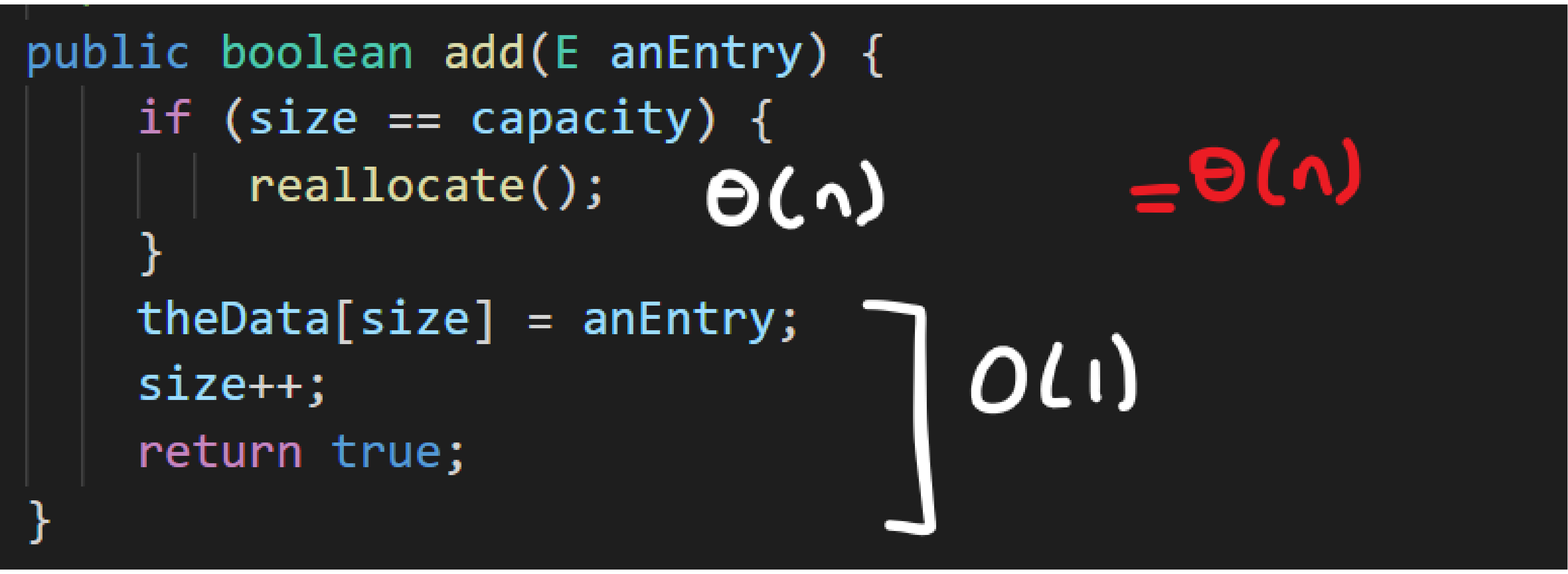
```
```

HybridList add method



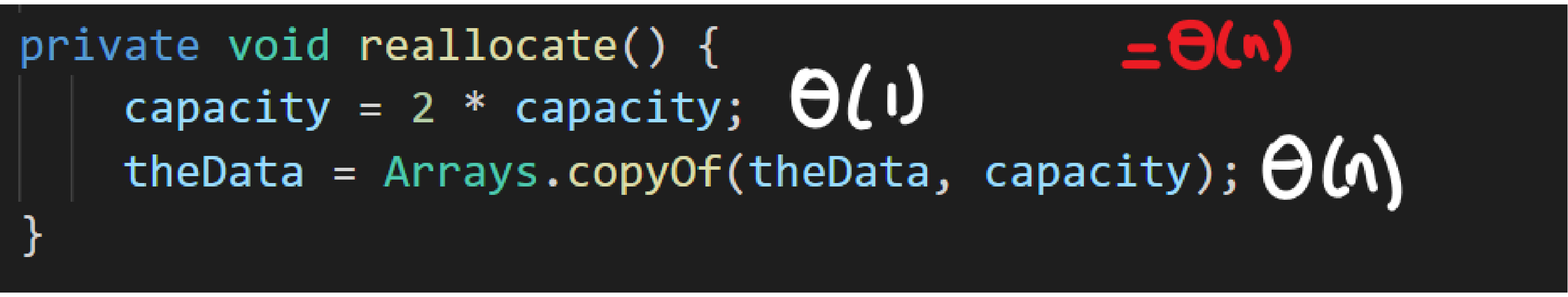
```
public boolean add(E e)
{
  KWArrayList<E> lastArr = this.linkedList.getLast();   Θ(1)

  if(lastArr == null)
  {
    this.linkedList.add(new KWArrayList<E>());   ] Θ(1)
    lastArr = this.linkedList.getLast();
  }else if(lastArr.size() == MAX_NUMBER)
  {
    this.linkedList.add(new KWArrayList<E>());   ] Θ(1)
    lastArr = this.linkedList.getLast();
  }

  lastArr.add(e);    ] Θ(1)
  this.size++;                          = Θ(1)
  return true;
}
```

KWArrayList add method



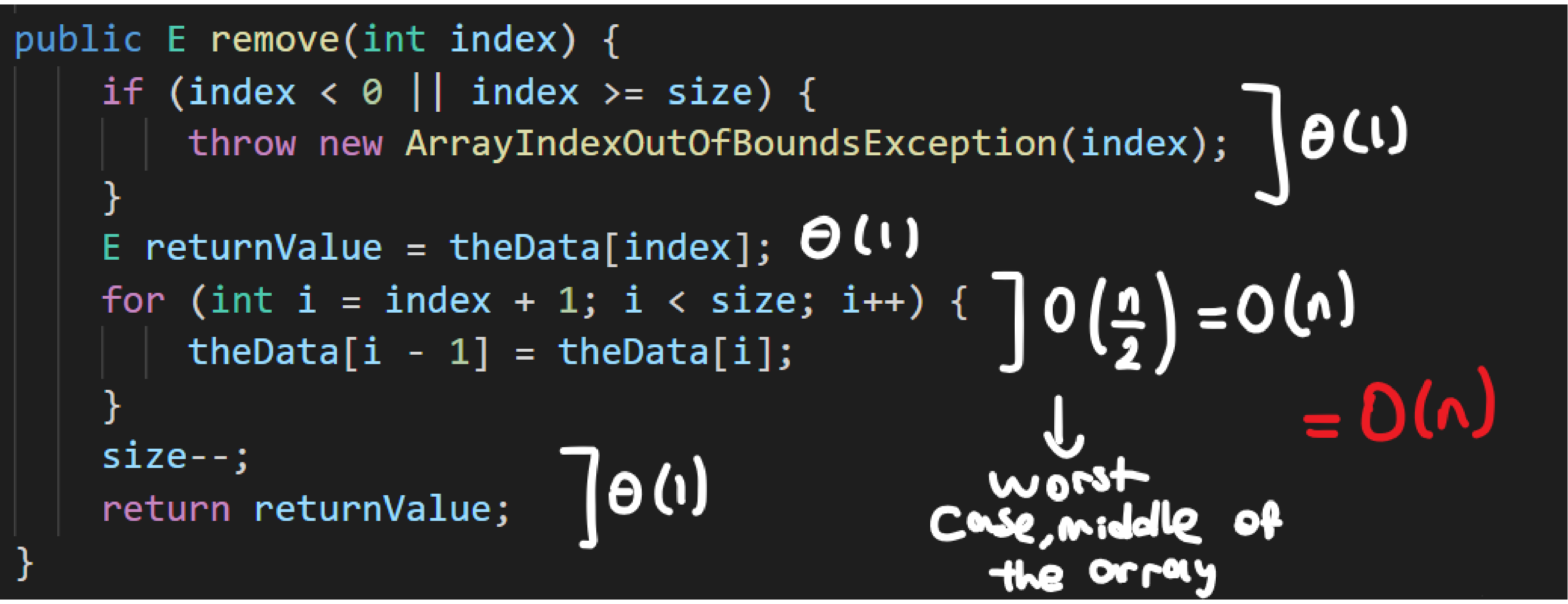
```
public boolean add(E anEntry) {
    if (size == capacity) {
        reallocate();    Θ(n)               = Θ(n)
    }
    theData[size] = anEntry;    ]
    size++;                     ] O(1)
    return true;                ]
}
```

KWArrayList reallocate method



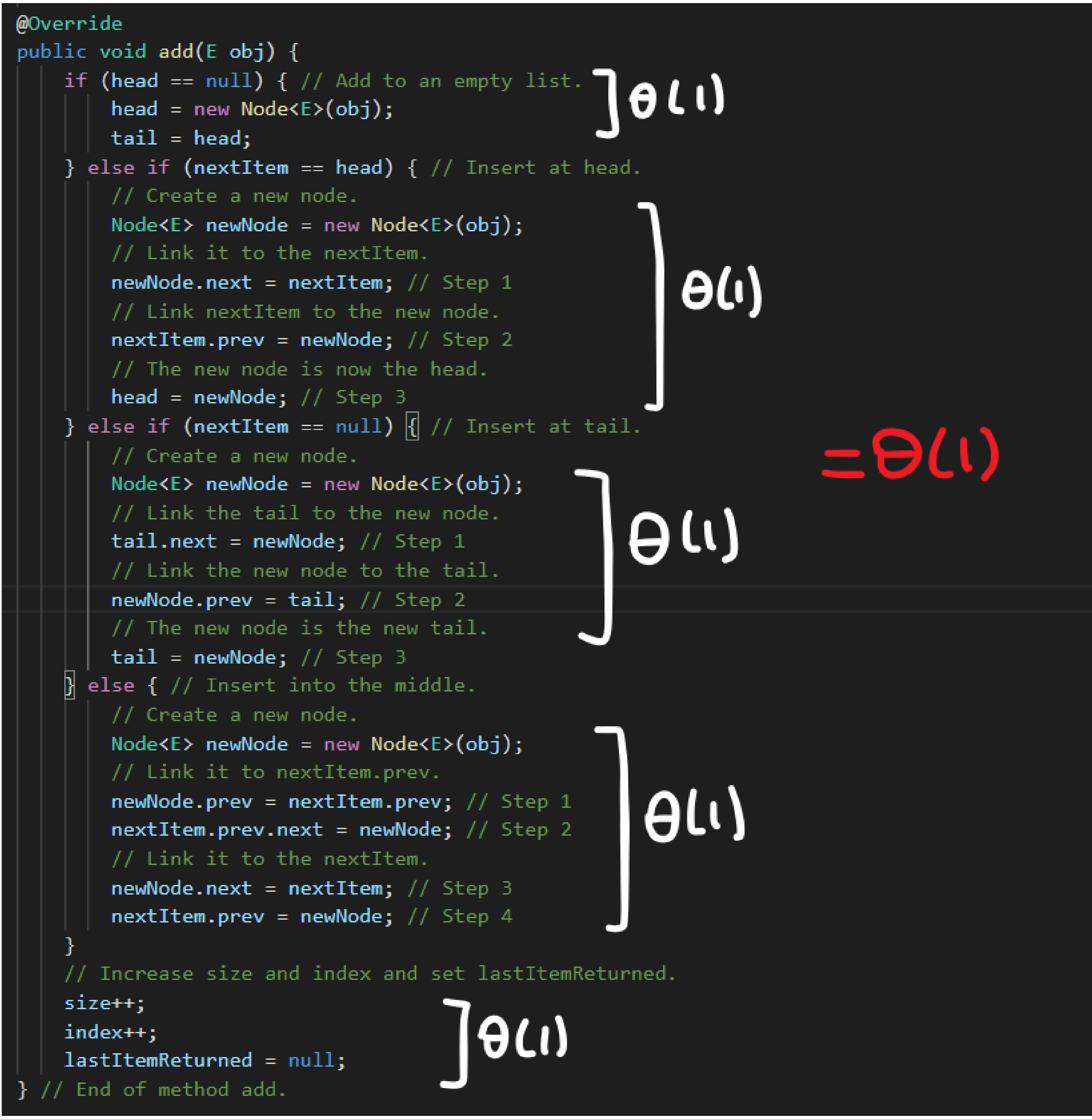
```
private void reallocate() {
    capacity = 2 * capacity;
    theData = Arrays.copyOf(theData, capacity);
}
```

KWArrayList remove method



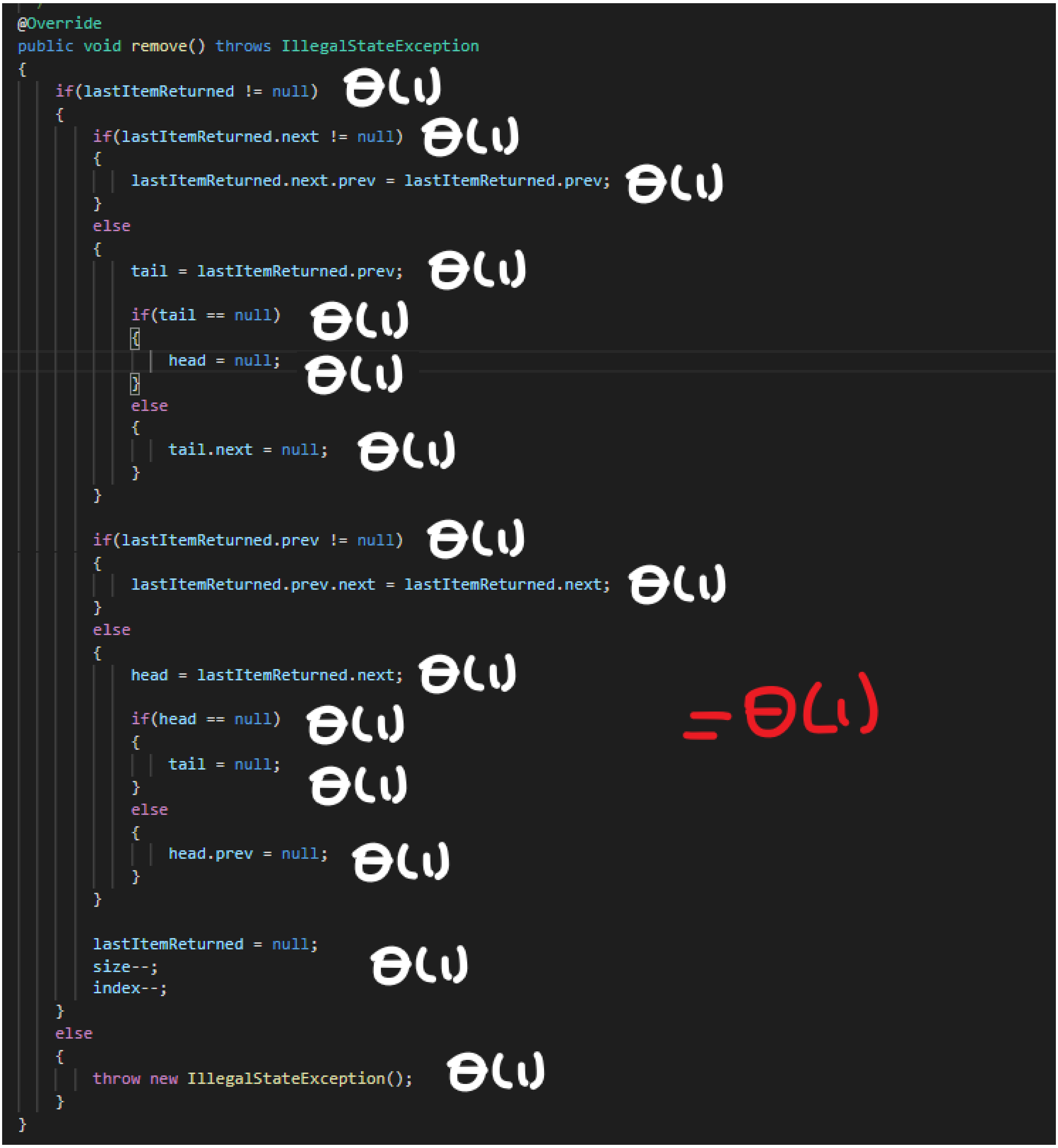
```
public E remove(int index) {
    if (index < 0 || index >= size) {
        throw new ArrayIndexOutOfBoundsException(index);
    }
    E returnValue = theData[index];
    for (int i = index + 1; i < size; i++) {
        theData[i - 1] = theData[i];
    }
    size--;
    return returnValue;
}
```

KWLinkedList add method



```
@Override
public void add(E obj) {
    if (head == null) { // Add to an empty list.
        head = new Node<E>(obj);
        tail = head;
    } else if (nextItem == head) { // Insert at head.
        // Create a new node.
        Node<E> newNode = new Node<E>(obj);
        // Link it to the nextItem.
        newNode.next = nextItem; // Step 1
        // Link nextItem to the new node.
        nextItem.prev = newNode; // Step 2
        // The new node is now the head.
        head = newNode; // Step 3
    } else if (nextItem == null) { // Insert at tail.
        // Create a new node.
        Node<E> newNode = new Node<E>(obj);
        // Link the tail to the new node.
        tail.next = newNode; // Step 1
        // Link the new node to the tail.
        newNode.prev = tail; // Step 2
        // The new node is the new tail.
        tail = newNode; // Step 3
    } else { // Insert into the middle.
        // Create a new node.
        Node<E> newNode = new Node<E>(obj);
        // Link it to nextItem.prev.
        newNode.prev = nextItem.prev; // Step 1
        nextItem.prev.next = newNode; // Step 2
        // Link it to the nextItem.
        newNode.next = nextItem; // Step 3
        nextItem.prev = newNode; // Step 4
    }
    // Increase size and index and set lastItemReturned.
    size++;
    index++;
    lastItemReturned = null;
} // End of method add.
```

KWLinkedList’s iterator’s remove method



```
@Override
public void remove() throws IllegalStateException
{
    if(lastItemReturned != null)  Θ(1)
    {
        if(lastItemReturned.next != null)  Θ(1)
        {
            lastItemReturned.next.prev = lastItemReturned.prev;  Θ(1)
        }
        else
        {
            tail = lastItemReturned.prev;  Θ(1)

            if(tail == null)  Θ(1)
            {
                head = null;  Θ(1)
            }
            else
            {
                tail.next = null;  Θ(1)
            }
        }

        if(lastItemReturned.prev != null)  Θ(1)
        {
            lastItemReturned.prev.next = lastItemReturned.next;  Θ(1)
        }
        else
        {
            head = lastItemReturned.next;  Θ(1)

            if(head == null)  Θ(1)
            {
                tail = null;  Θ(1)
            }
            else
            {
                head.prev = null;  Θ(1)
            }
        }

        lastItemReturned = null;
        size--;                      Θ(1)
        index--;
    }
    else
    {
        throw new IllegalStateException();  Θ(1)
    }
}
```

= Θ(1)