CSE 222 Assignment 3

Part 2

Searching Product ;

```
public boolean _searchProduct(Company company, Branch selected, Product product
    if(company.getBranches().indexOf(selected) !=-1){ ) O(n)   int choice){
        return(selected.getProducts().check(choice, product)); ) O(1)
    }
    return false; ⇒ O(1)
}
```

O(1) O(n) O(1) O(1)

	cost	Frequency	Total
1	3	n	3n
2	3	1	3
3	1	1	1

$T(n) = 3n+3$ $O(n)$

$T(n) = 3n+1$ $O(n)$

Add Product;

```
public void _addProduct(Company company, Branch selected, Product product, intchoi
    if(company.getBranches().indexOf(selected) !=-1){ ) O(n)
        selected.getProducts().add(choice, product); } O(1)
    }
}
```

O(1) O(n) O(1) O(1)

	cost	Frequency	Total
1	3	n	3n
2	2	1	2
3			

$T(n) = 3n+2$

$\Theta(n) = 3$

Remove Product

```
public void removeProduct(Company company, Branch selected, Product product, int choice){
    if(company.getBranches().indexOf(selected) != -1){ } O(n)
        O(1)   O(n)
        selected.getProducts().remove(choice, product); }O(n)
                 O(1)          O(n)
    }
}
```

	cost	Frequency	Total
1	3	(n+1)	3n+1
2	2	(n+1)	2n+1

$T(n) = 5n+1$

$O(n)$

Querying Product

```
public String inquireProduct(Company company, Branch selected, Product product, int choice)
                                    O(n)
    if(company.getBranches().indexOf(selected) != -1){ O(n)
        O(1)
        return selected.getProducts().toString(choice, product); O(n)
                        O(1)          O(n)
    }
    return "There is no branch";
}
```

	cost	Frequency	Total
1	3	(n+1)	3n+3
2	2	n	2n
			5n+3

$T(n) = 5n+3$

$O(n)$

AddBranch

```
Public void _addBranch(Company company, Branc branch){
    company.setBranch_number(company.getBranch_number()+1); O(1)      Ω(1)
    company.getBranches().add(branch); O(1)                           O(1)
    System.out.println(branch.getName() + " has been added"); O(1)
}
```

Renove Branch

```
public String _renoveBranch(Company company, Branch selectedBranch){
    Iterator<Branch> it = company.getBranches().iterator(); ⇒ O(1)

    String branchname = selectedBranch.getName(); ⇒O(1)
    int count=0; ⇒ O(1)
    while(it.hasNext()){
        if(it.next() == selectedBranch){⇒ O(1)
            System.out.println(count); ⇒ O(1)
            company.getBranches().renove(count);⇒O(n)   Ω(1)  O(n)   O(n²)   O(n²)
        }
        count++;
    }
    company.setBranch_number(company.getBranch_number()-1); ⇒O(1)      O(n²)
    return branchName; ⇒ O(1)
}
```

Add Emplyee

```
public void _addBEmplyee(Company company, BranchEmplyee employee){
    company.SetEmployee_number(company.getEmplyee_number()+1); ⇒ O(1)
    company.getEmplyee().add(employee); ⇒ O(1)
    System.out.println(employee.name + " has been added."); ⇒O(1)      O(1)
}
```

Remove Employee

```
public void removeBEmployee(Company company, BranchEmployee selectedEmployee){
    try{                                                          O(n²)
        company.getEmployee().remove(company.getEmployee().indexOf(selectedEmployee));
        company.setEmployee_number(company.getEmployee_number()-1);
    }catch(Exception e){
        System.out.println("There is no existing employee");
    }
}
```

O(1) + O(n²) + O(n) + O(1) + O(1) = **O(n²)**

New Subscription

```
public void newSubscription(Company company, Customer newCustomer){
    company.setCustomer_number(company.getCustomer_number()+1);
    company.getCustomers().add(newCustomer);
}
```

O(1) + O(1) + O(1) = **O(1)**

Update Previous Order

```
public void updatePreviousOrder(Company company, Customer customer, Product product, int choice){
    int index = company.getCustomers().indexOf(customer);
    if(index != -1){
        company.getCustomers().get(index).getProducts().add(choice, product);
    }
}
```

O(1) + O(1) + O(1) + O(1) + O(1) + O(1) = **O(1)**

Access Previous Order

```
public String _accessPrevious Order(Company company, Customer custoer, int x, int y){
    int index = company.getCustomers().indexOf(customer); => O(1)
    if(index != -1){ => O(1)
        return company.getCustomers().get(index).getProducts().previous(x-1, y-1);
                         O(1)           O(1)        O(1)            O(n)
    }
    return "";                                          Θ(n)  O(n)  Ω(1)
}
```

Search Product

```
public boolean _searchProduct(Company company, Branch selected, Product product, int choice){
    if(company.getBranches().indexOf(selected) != -1){ => O(1)
        return (selected.getProducts().check(choice, product));     O(n)   Ω(1)
                           O(1)          O(n)                        Θ(n)
    }
    return false;
}
```

Customer Class

Order

```
public void _order(Company company, Branch selected, Product product){
    if(company.getBranches().indexOf(selected) != -1){ => O(n)
        if(selected.getProducts().check(product.getChoice(), product)){ => O(n)
            this.products.add(product.getChoice(), product); O(1)
            selected.getProducts().remove(product.getChoice(), product); => O(n)
            order_product_number ++;            T(n) = 3n+1
                                                O(n)   Ω(n)
            }
        }
    }
}
```

Product Class

Add

```
public boolean add(int choice, Product product){
    this.x = product.x;  ⇒ O(1)
    this.y = product.y;  ⇒ O(1)
    this.choice = choice; ⇒ O(1)
    switch(choice){                                              O(1)
      case 1:
          return chair.add(Product.x, product.y, product.cant); ⇒ O(1)
      case 2:
          return desk.add(product.x, product.y, product.cant);
      case 3:
      case 4:
      case 5:
    }
    return false;
}
```

Check

```
public boolean check(int choice, Product product){
   switch (choice){
        case 1:                   O(n)                                      O(n)
              return chair.sumOf(product.x, product.y) > product.chair.sumOf(product.x, product.y)
        case 2:
        case 3:
        case 4:                                                       O(n)
        case 5:
   }
   return false
}
```

```
Public String previous(int choice, int x, int y){
    switch (choice){
        case 1:
            return chair.toString(x,y); =>O(1)
        case 2:
        case 3:                                    O(1)
        case 4:
        case 5:
    }
    return "";
}
```

Remove Last Element

```
public int removeLast(){
    switch (choice){
        case 1:
            return chair.lastProduct(x,y); =>O(1)
        case 2:
        case 3:                                    O(1)
        case 4:
        case 5:
    }
    return -1;
}
```

Remove

```
switch (choice){
    case 1:
        if(chair.lastGet(product.x, product.y) > product.cant){   // O(1), O(1)
            removeLast(); =>O(1)
            chair.add(product.x, product.y, product.cant); ->O(1)
            return true;
        }
        product.cant-= removeLast(); =>O(1)
        return remove(product.chaire, product); =>O(1)
                                                   } O(n)
    case 2:
    case 3:
    case 4:
    case 5:
} return false;
```