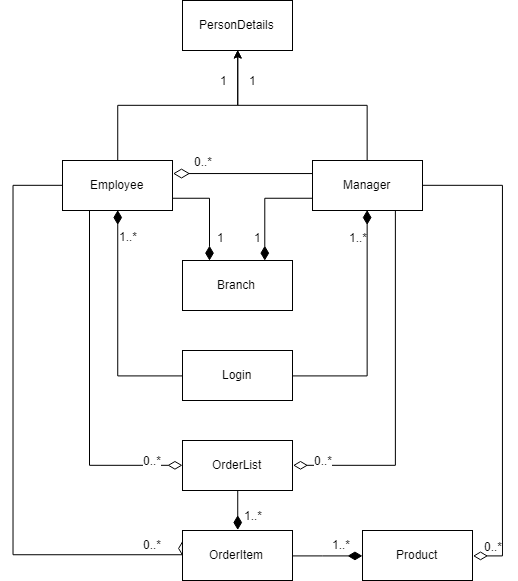
****

**PersonDetails Employee**

|  |
| --- |
| PersonDetails |
| #firstName: String  #lastName: String  #gender: char  #phoneNo: String  #email: String  #icNo: String |
| +PersonDetails()  +PersonDetails(personDetails: PersonDetails)  +PersonDetails(firstName: String, lastName: String, gender: char, phoneNo: String, email: String, icNo: String)  +getFirstName(): String  +getLastName(): String  +getGender(): char  +getPhoneNo: String  +getEmail(): String  +getIcNo(): String  +setFirstName(firstName: String): void  +setLastName(lastName: String): void  +setGender(gender: char): void  +setPhoneNo(phoneNo: String): void  +setEmail(email: String): void  +setIcNo(icNo: String): void  +toString(): String |

|  |
| --- |
| Employee |
| -nextEmployeeID: int  -jobTitle: String  -employeeID: String  -password: String  -branch: Branch |
| +Employee()  +Employee(personDetails: PersonDetails, jobTitle: String, password: String, branch: Branch)  +modifyOrderList(productCode: String, orderItem: ArrayList<OrderItem>, listNo: int): OrderList  +payment(orderList: OrderList): void  +displayProduct(orderItem: ArrayList<OrderItem>): int  +displayTransactionHistory(orderLists: ArrayList<OrderList>): void  +getNextEmployeeID(): int  +getEmployeeID(): String  +getJobTitle(): String  +getPassword(): String  +getBranch(): Branch  +setJobTitle(jobTitle: String): void  +setPassword(password: String): void  +setBranch(branch: Branch): void  +toString(): String |

|  |
| --- |
| Login |
| -username: String  -password: String  -i: int |
| +currentTime(): String  +employeeLogin(employeesArray: ArrayList<Employee>): boolean  +managerLogin(manager: Manager)  +currentTime(): String  +getUsername(): String  +getPassword(): String  +getIndex(): int  +setUsername(username: String): void  +setPassword(password: String): void |

**Branch Login**

|  |
| --- |
| Branch |
| -nextBranchId: int  -branchId: String  -branchName: String  -manager: Manager |
| +Branch()  +Branch(branchName: String, manager:Manager)  +getBranchId(): String  +getBranchName(): String  +getManager: Manager  +setBranchName(branchName: String): void  +setManager(manager: Manager): void  +toString(): String |

|  |
| --- |
| Manager |
| -nextManagerID: int  -managerID: String  -password: String  -jobTitle: String |
| +Manager()  +Manager(personDetails: PersonDetails, password: String, jobTitle: String)  +modifyProduct(product: ArrayList<Product>): boolean  +editProduct(i: int, product: ArrayList<Product>): boolean  +addProduct(productId: String, product: ArrayList<Product>): boolean  +modifyStaff(employees: ArrayList<Employee>): boolean  +editStaff(i: int, staff: ArrayList<Employee>): boolean  +addStaff(staff: ArrayList<Employee): boolean  +dailyReport(orderList: ArrayList<OrderList>, products: ArrayList<Product>): void  +getNextManagerID(): int  +getManagerID(): String  +getPassword(): String  +setPassword(password: String): void  +setJobTitle(jobTitle: String): void  +toString(): String  -validGender(input: char): boolean  -validPhoneNo(input: String): boolean  -validEmail(input: String): boolean  -promptChoice(max: int): int |

|  |
| --- |
| OrderList |
| -orderNo: String  -nextOrderNo: int  -orderItem: ArrayList<OrderItem>  -totalAmount: double  -itemCount: int  -amount: double  -formattedDate: String |
| +OrderList()  +addOrderItem(item: OrderItem): boolean  +editQuantity(list: int, quantity: int): void  +receipt(paid: boolean, amount: double): void  +setOrderNo(orderNo: String): void  +setOrderNo(nextOrderNo: int): void  +setOrderItem(orderItem: ArrayList<OrderItem>): void  +setTotalAmount(totalAmount: double): void  +setItemCount(itemCount: int): void  +setAmount(amount:double): void  +setFormattedDate(formattedDate: String): void  +getOrderNo(): String  +getNextOrderNo(): int  +getOrderItem(): ArrayList<OrderItem>  +getTotalAmount(): double  +getItemCount(): int  +getFormattedDate(): String  +getAmount(): double |

**Manager OrderList**

|  |
| --- |
| OrderItem |
| -product: Product  -amount: double  -quantity: int |
| +OrderItem()  +OrderItem(product: Product)  +getProduct(): Product  +getAmount(): double  +getQuantity(): int  +setProduct(product: Product)  +setAmount(amount: double)  +setQuantity(quantity: int)  +stockOut(quantity: int): boolean  +toString(): String |

|  |
| --- |
| Product |
| -prodId: String  -prodName: String  -prodType: String  -stockQuantity: int  -price: double  -nextProdId: int |
| +Product(prodName: String, prodType: String, price: double, stockQuantity: int)  +Product(prodName: String, prodType: String, price: double)  +Product(prodName: String, price: double, prodType: String)  +Product(prodName: String, price: double)  +setNextProdId(nextProdId: int): void  +setProdName(prodName: String): void  +setProdType(prodType: String): void  +setStockQauntity(stockQuantity: int): void  +setPrice(price: double): void  +getProdId(): String  +getProdName(): String  +getProdType(): String  +getStockQuantity(): int  +getPrice(): double  +toString(): String |

**OrderItem Product**

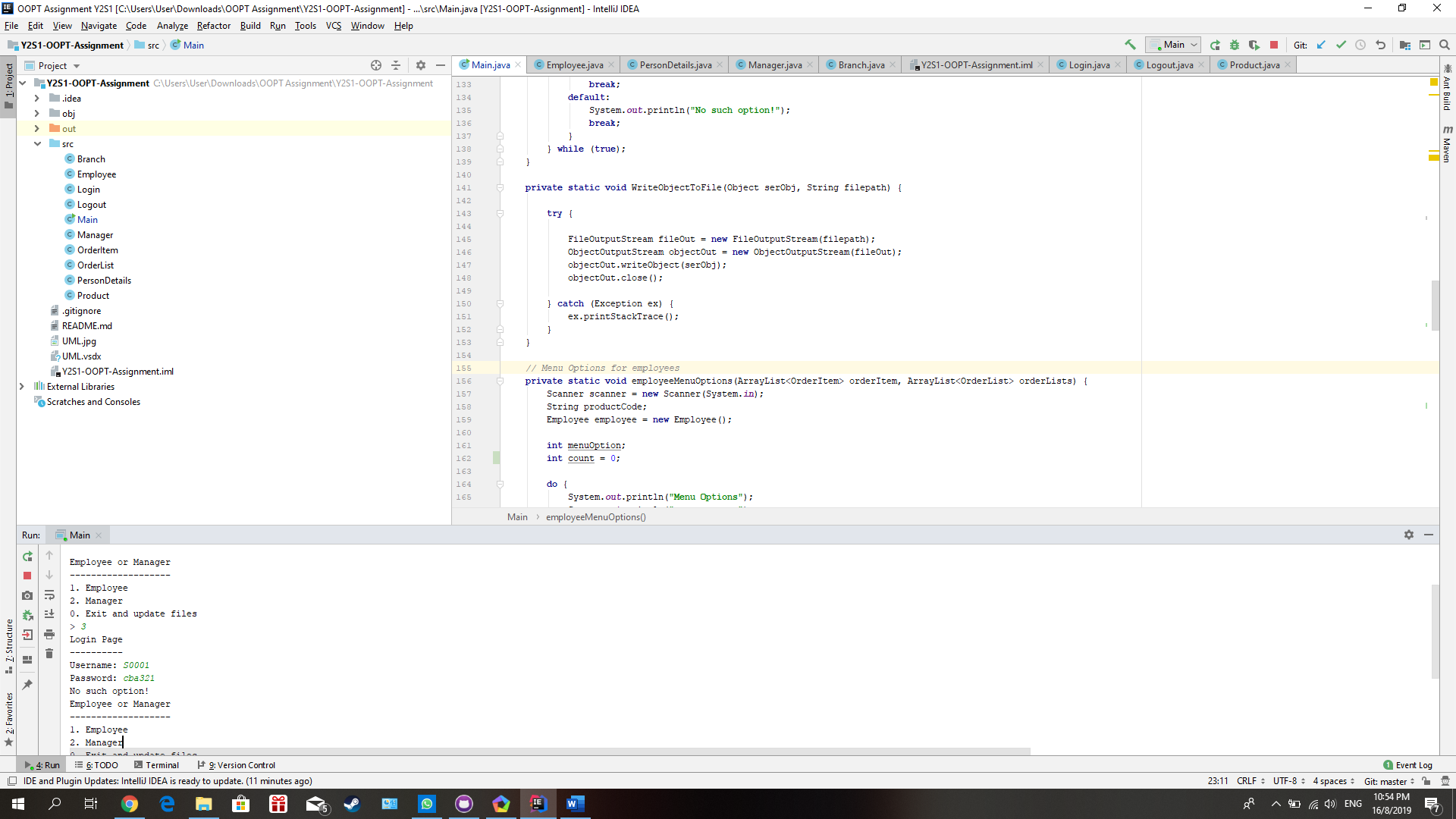
**Assignment Idea**

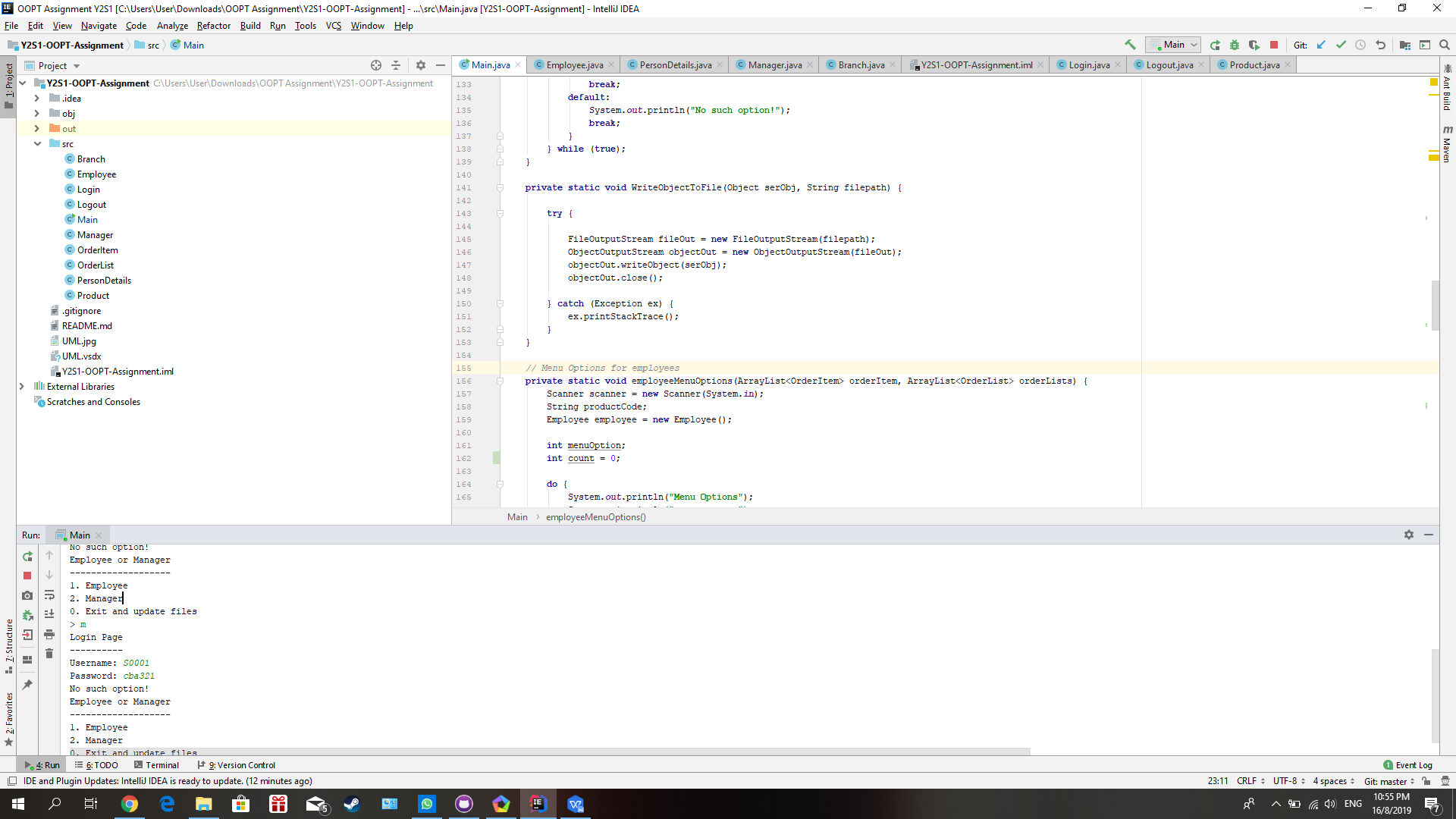
For the AACS2204 assignment, we are doing a Point of Sale (POS) system on a pet store. We assume that only staff will be using the POS for daily transactions, and managers only can add and edit staff and product details. The POS is designed in console, a user will have to type in their responds but all the responds are validated before entering to the system. If the responds they enter is not valid, they will get an error message that will also hint the user to enter the correct value. Our pet store sells various of pets related items such as dog food, and pet toys.

Our POS system includes some extra features compare to the POS system in the market such as adding and editing staff and product details. The system will update all the data when a user exits the system to make sure the modified staff and product will be updated. All the data are saved into binary files.

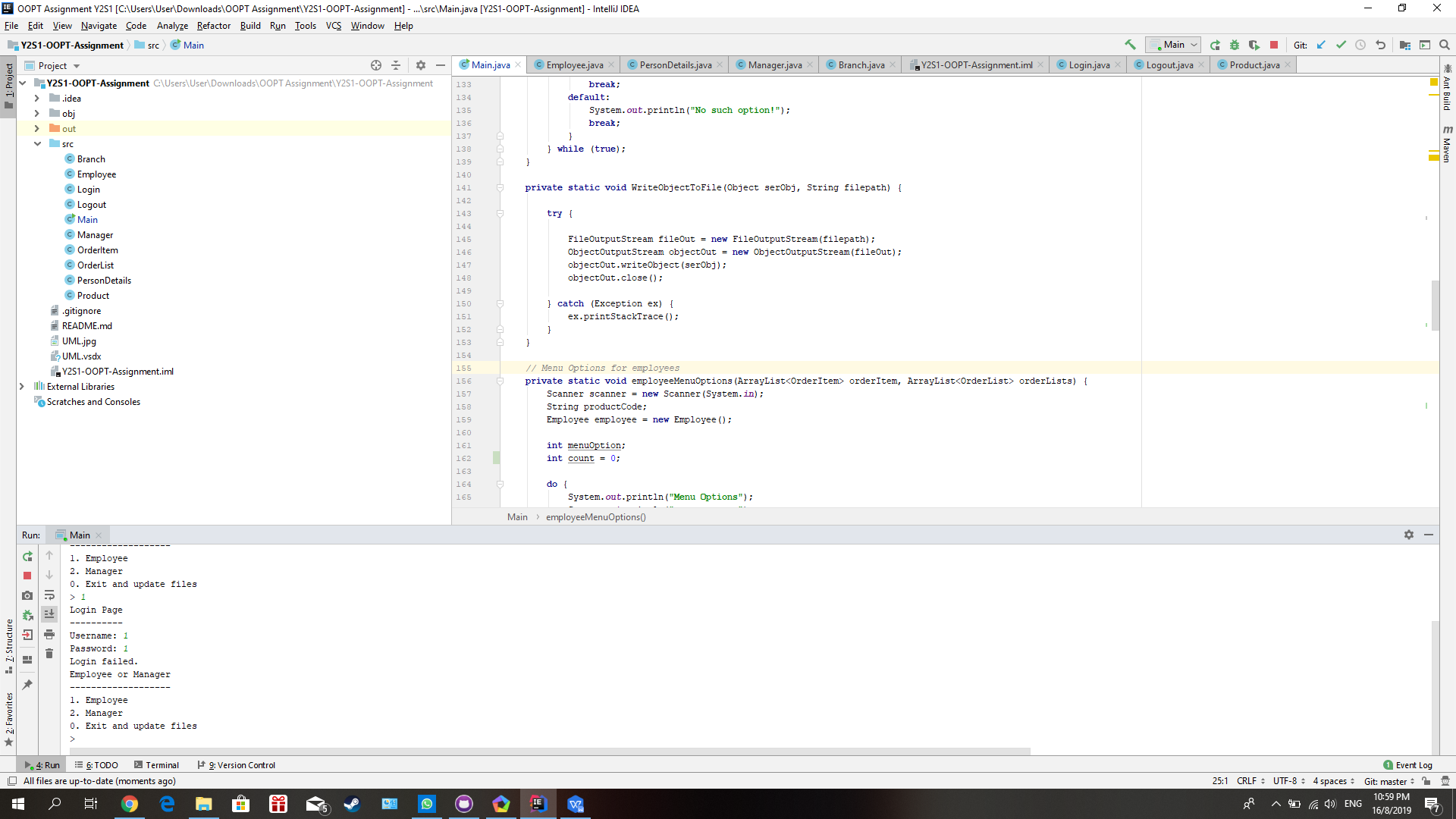
Our POS system do not support input and output other than a keyboard and screen. The POS system do not support input from a barcode scanner but the program is expandable for various type of inputs. The program is also expandable for graphical interface.

1. **Login**

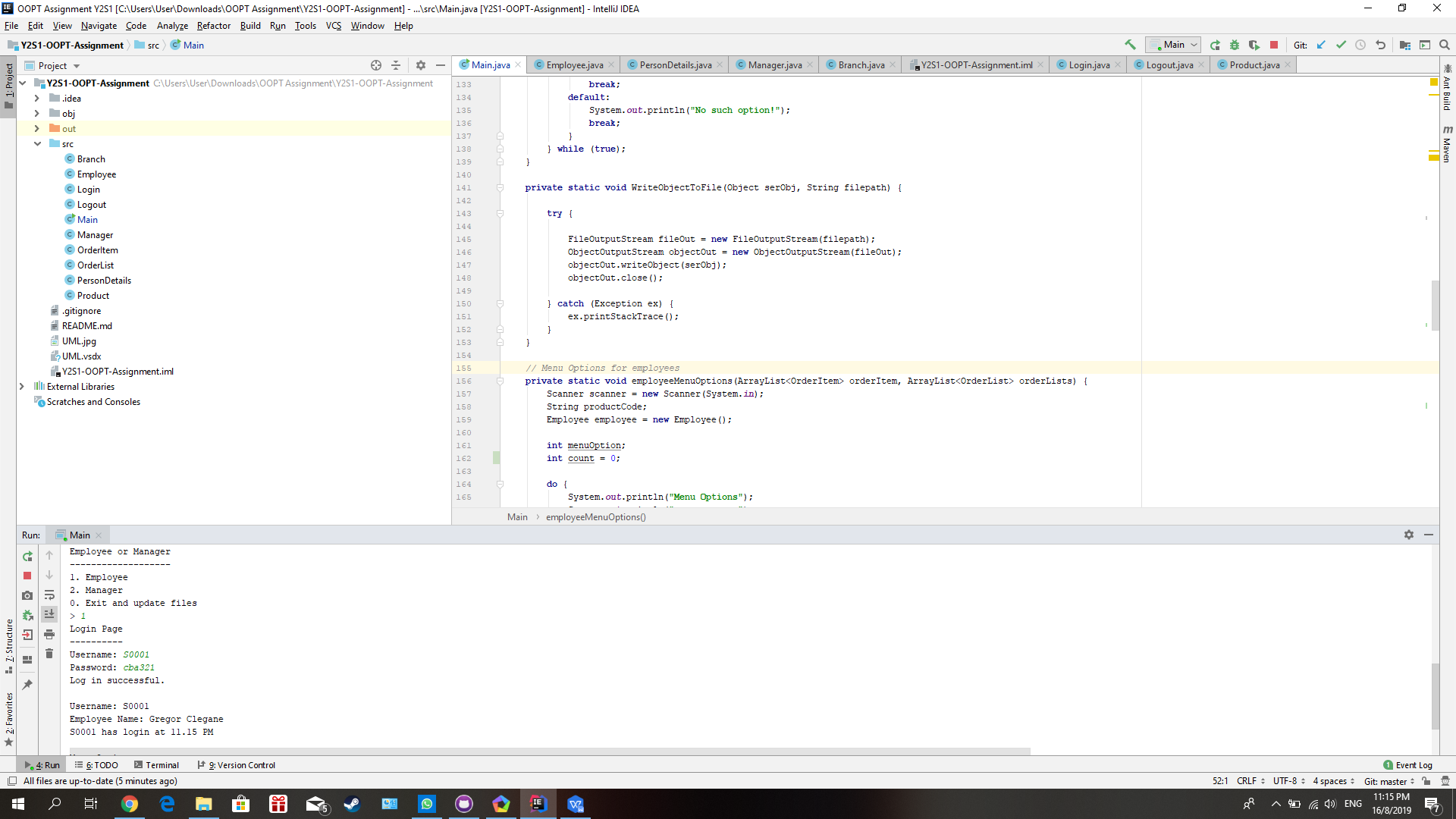




Enter invalid number or alphabet will display an error message such as no such option to user.

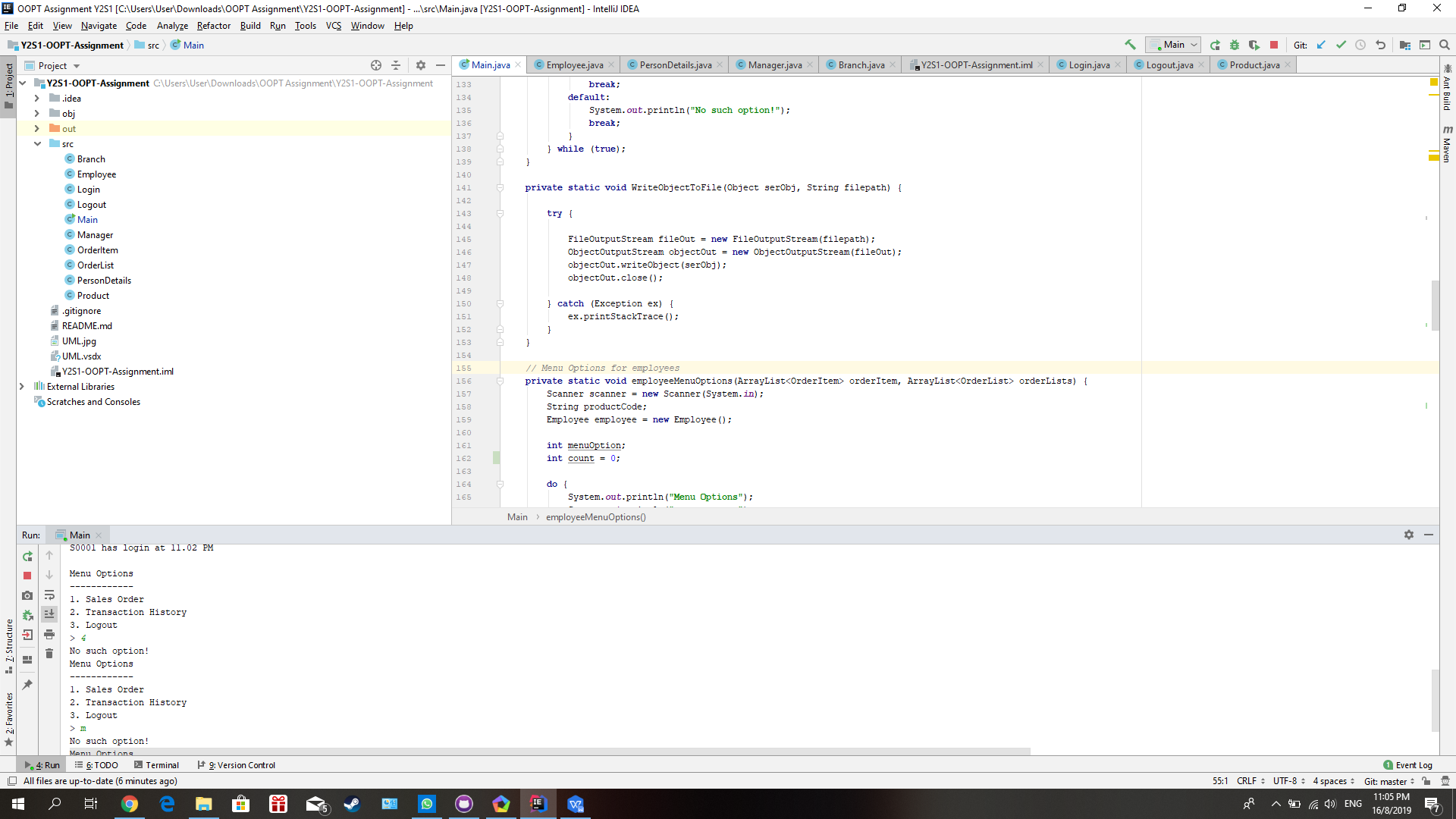


Enter invalid username or password in login page will display the “login failed.” message to user.

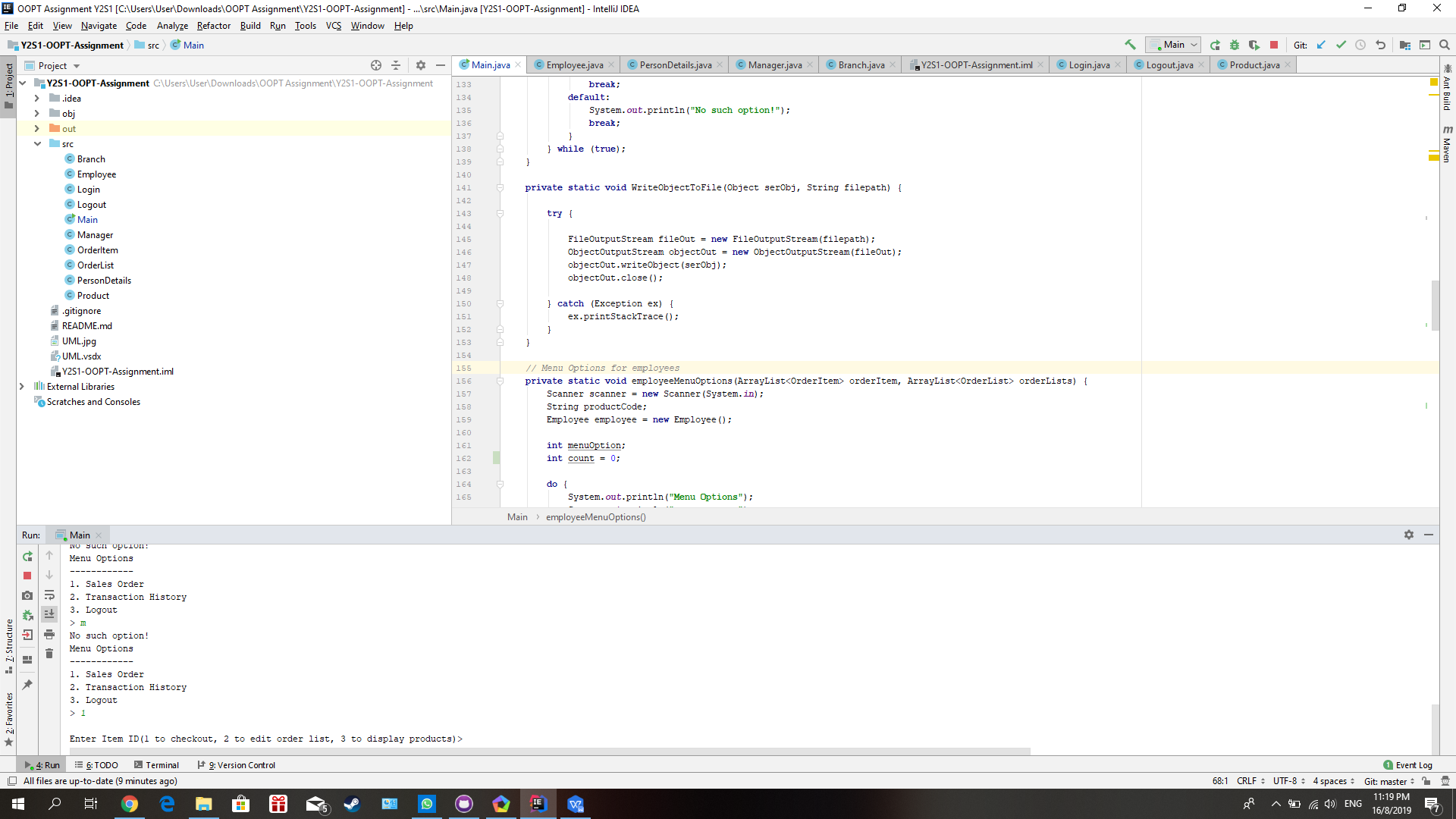


This is the example of entering valid username and password and it will display “Log in successful.” when the user enter correct username and password. It will also display the username, name of the user and the login time once the user has login.

1. **Menu Options**

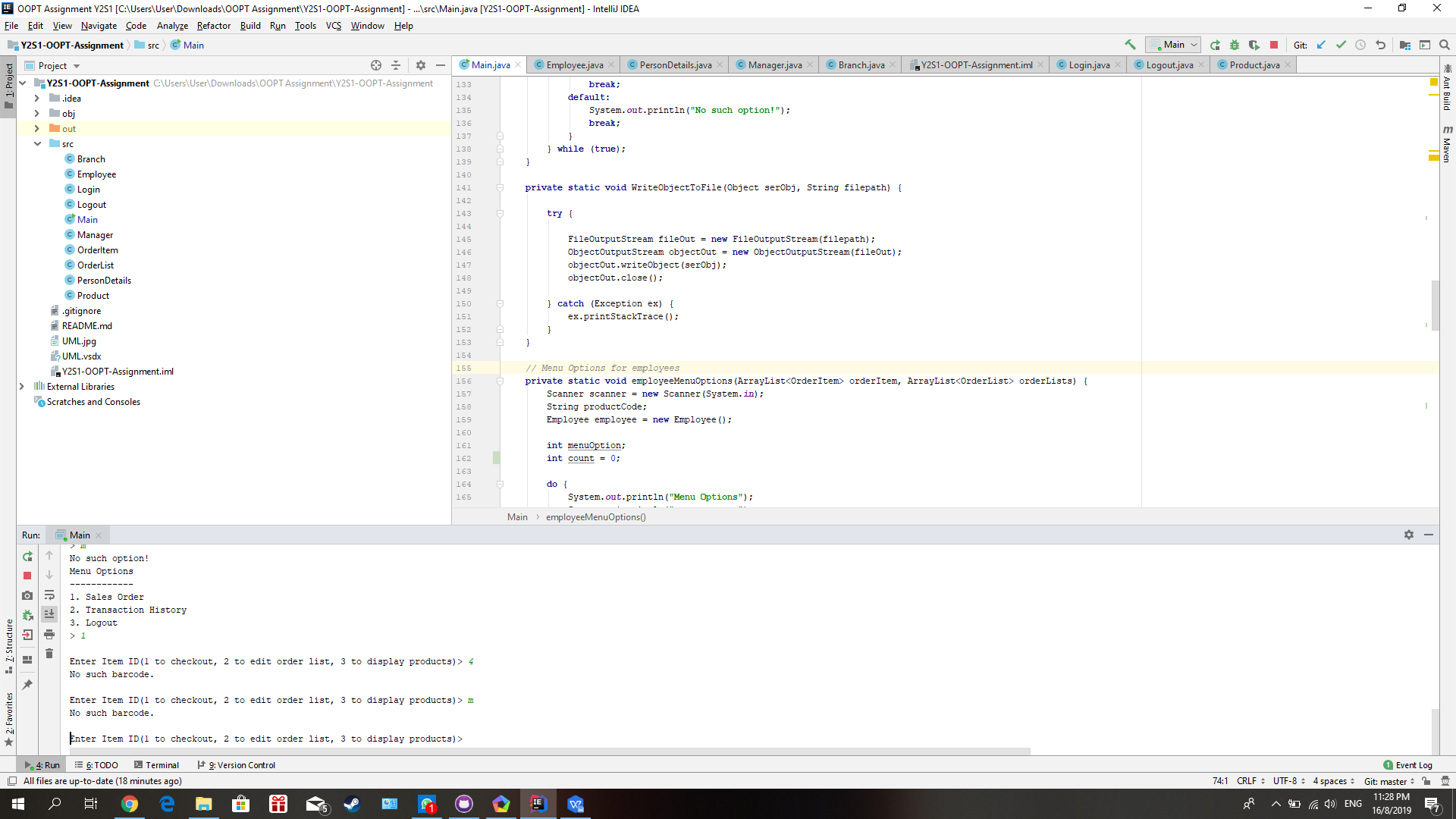


In the menu options, invalid input such as invalid number or alphabet is not available and if the user enter a invalid input the system will display an error message to the user and display again the menu options to the user.

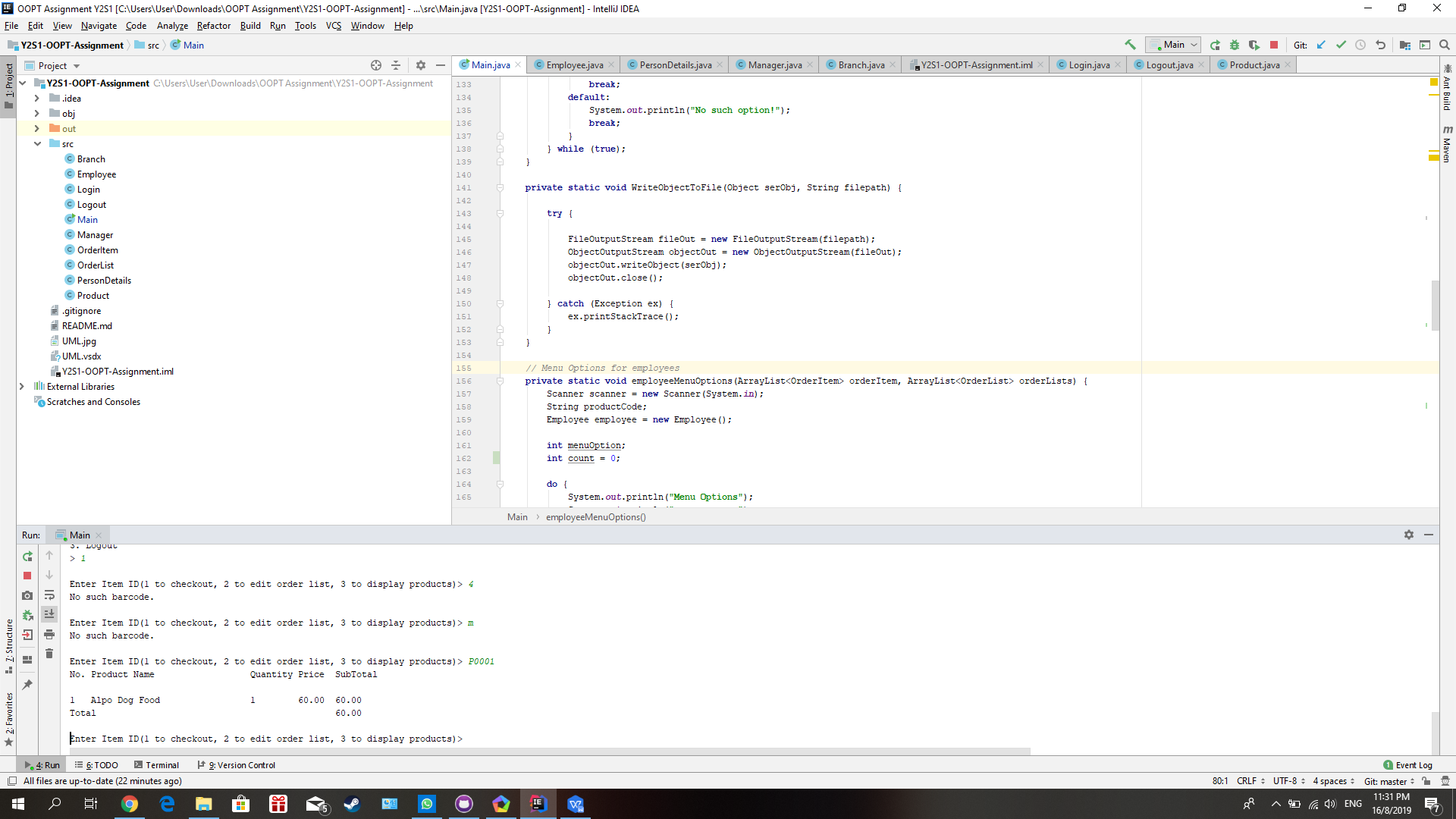


This is the example of entering the valid input.

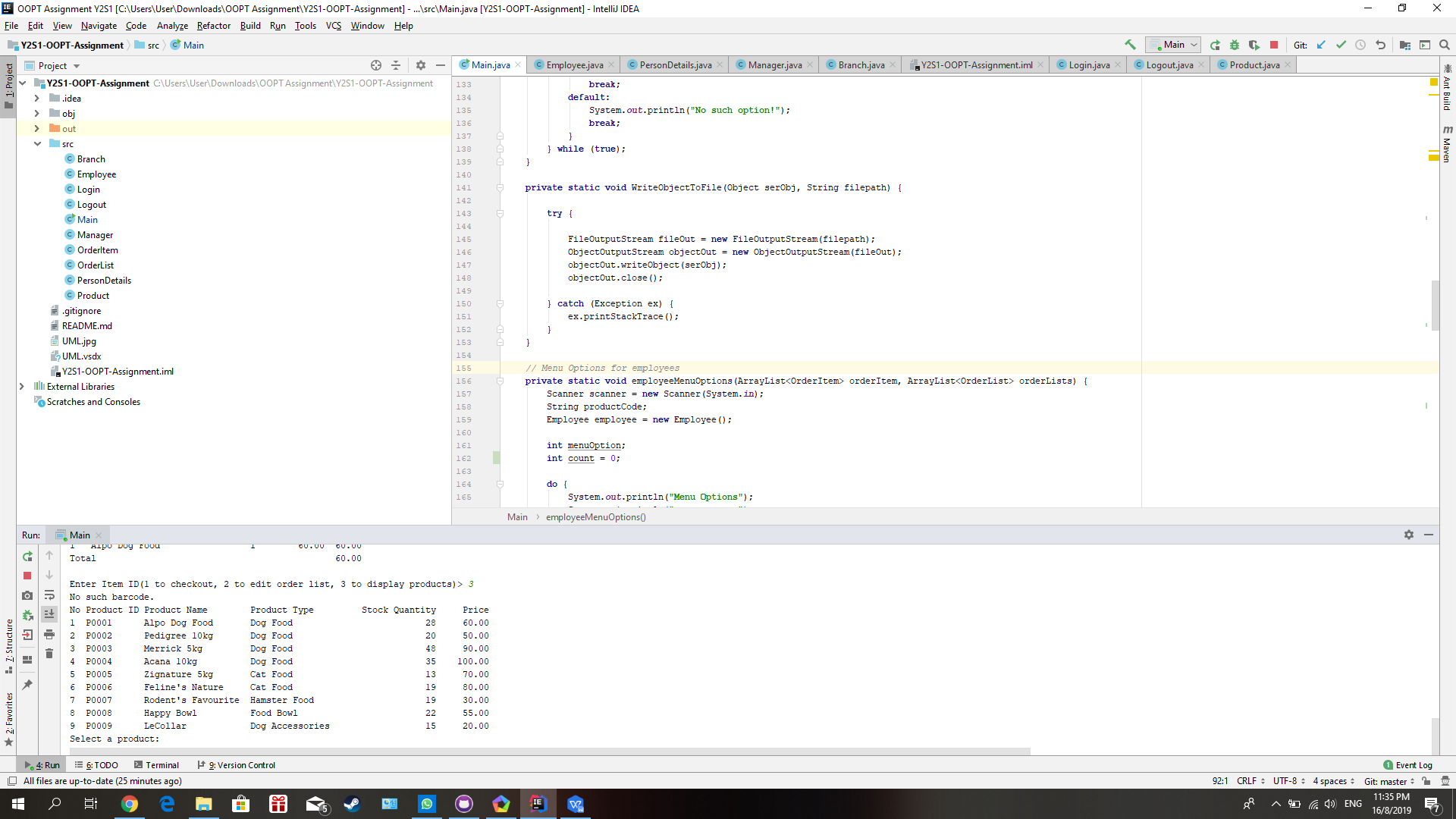
1. **Sales Order**

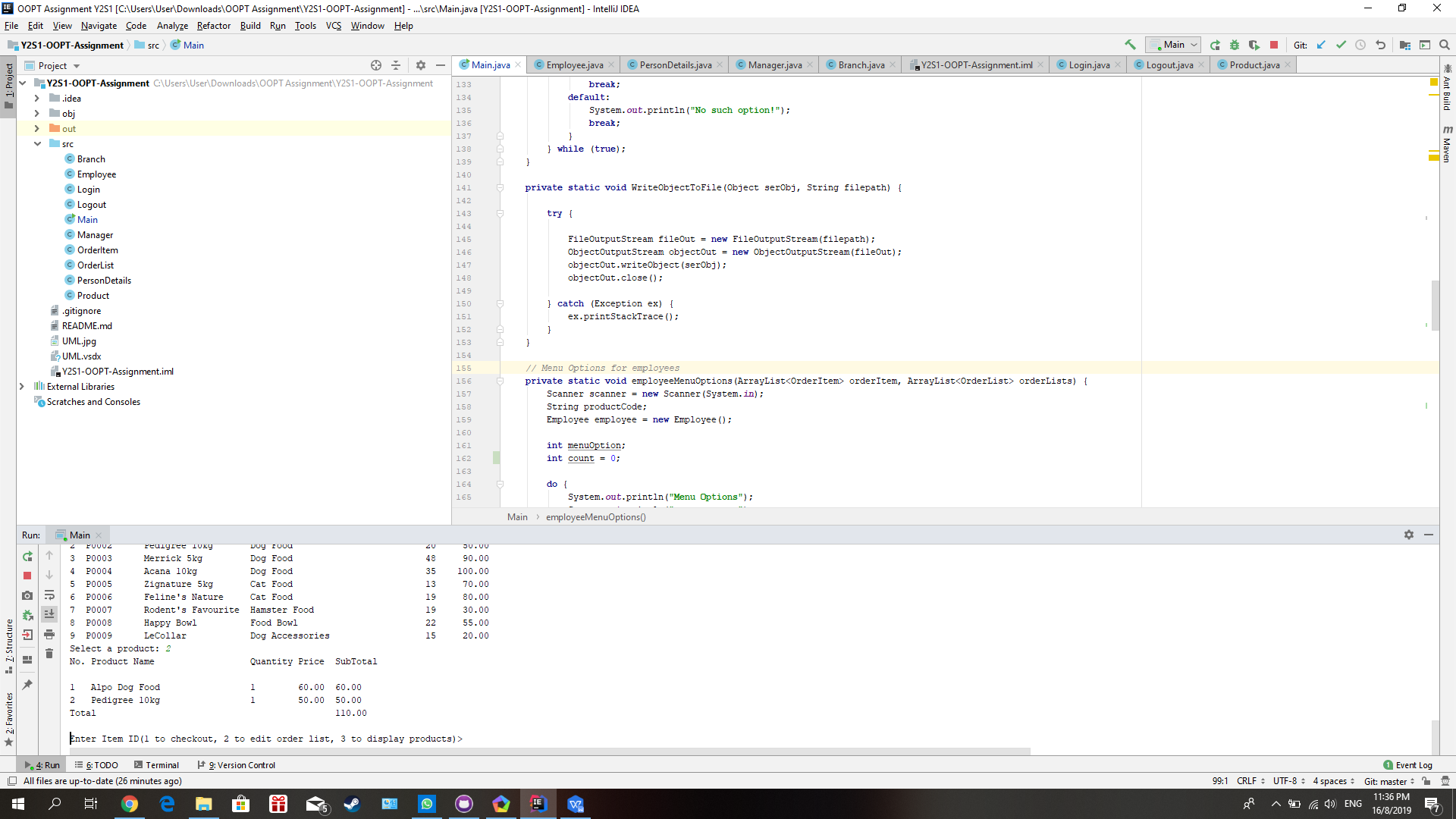


Invalid input such as invalid number and alphabet shown in the picture is not available and will display error message such as “No such barcode.” and will repeat until the user enter valid input.

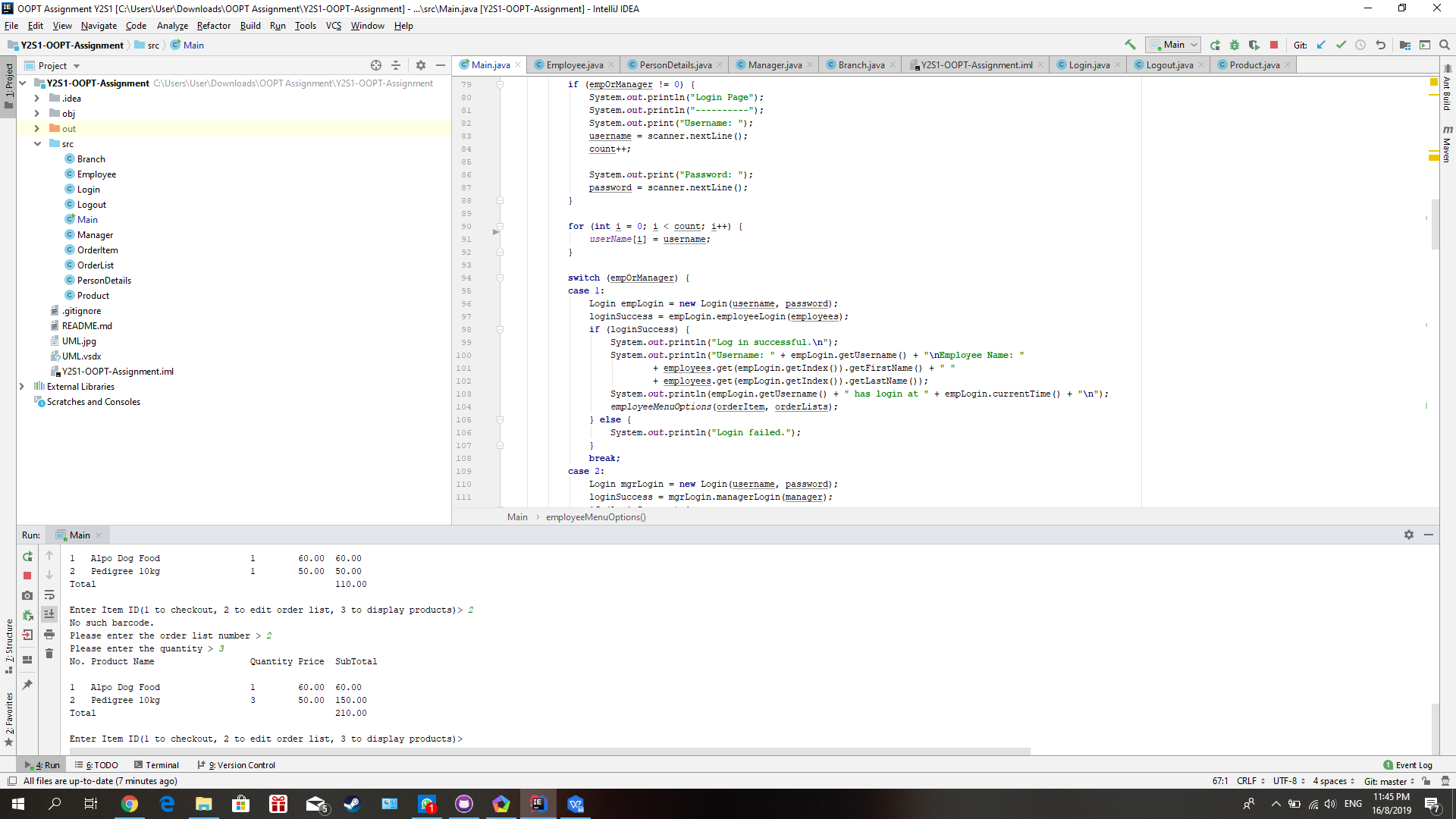


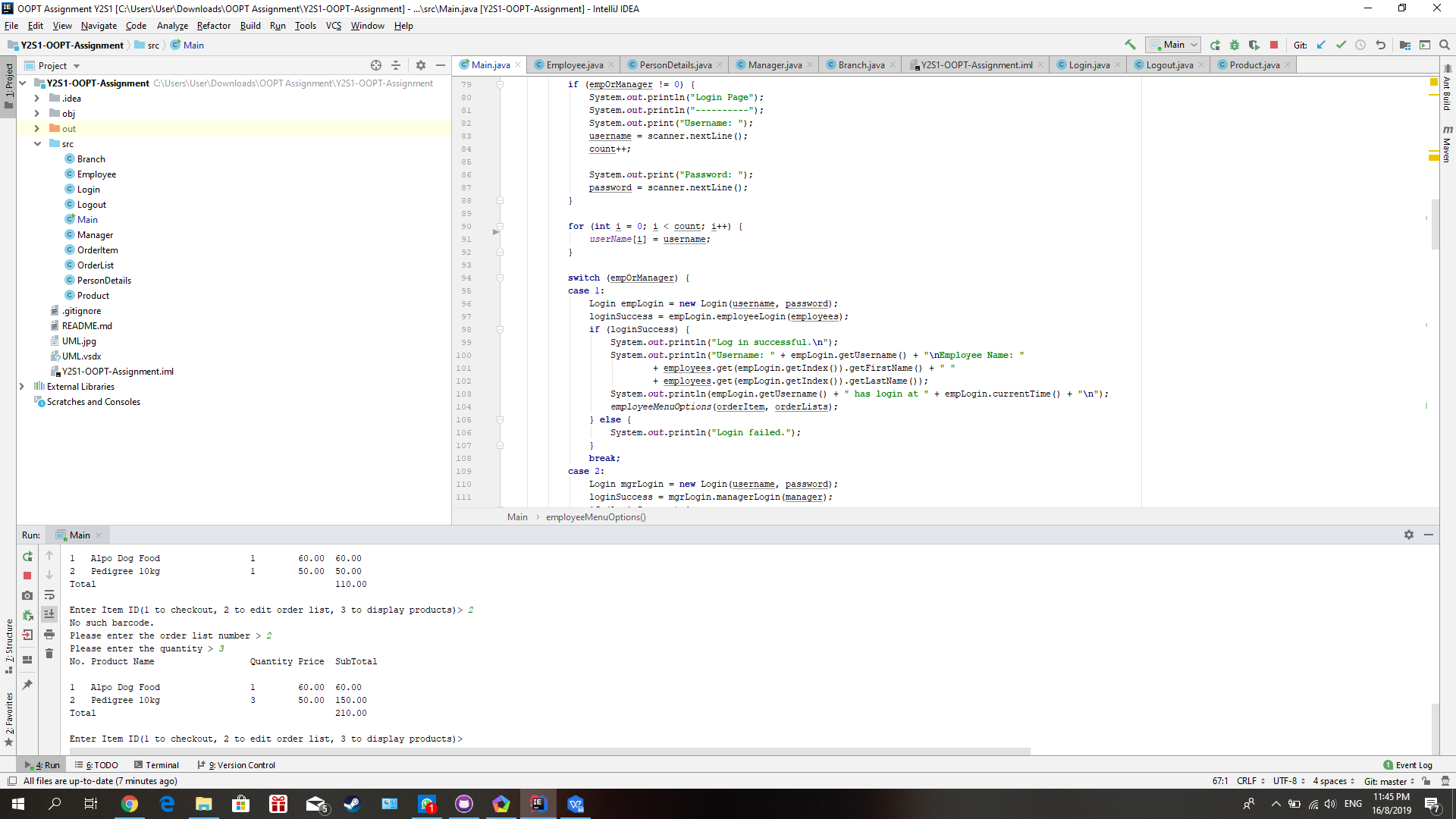
User can enter the item ID such as P0001 to purchase the item as the picture shown.



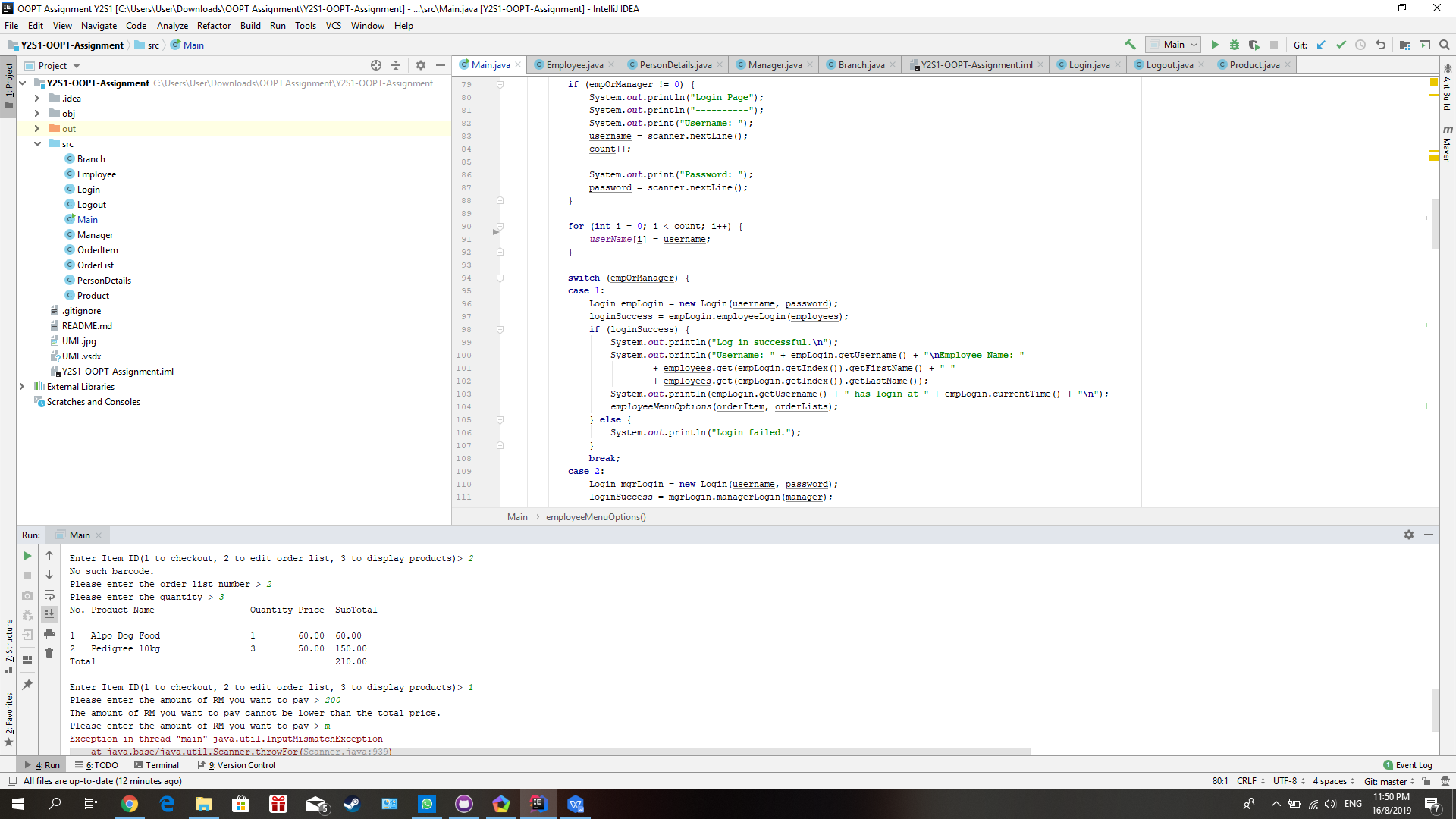


User can enter the number 3 to display the products list and enter the number of the product to purchase the item as the picture shown.

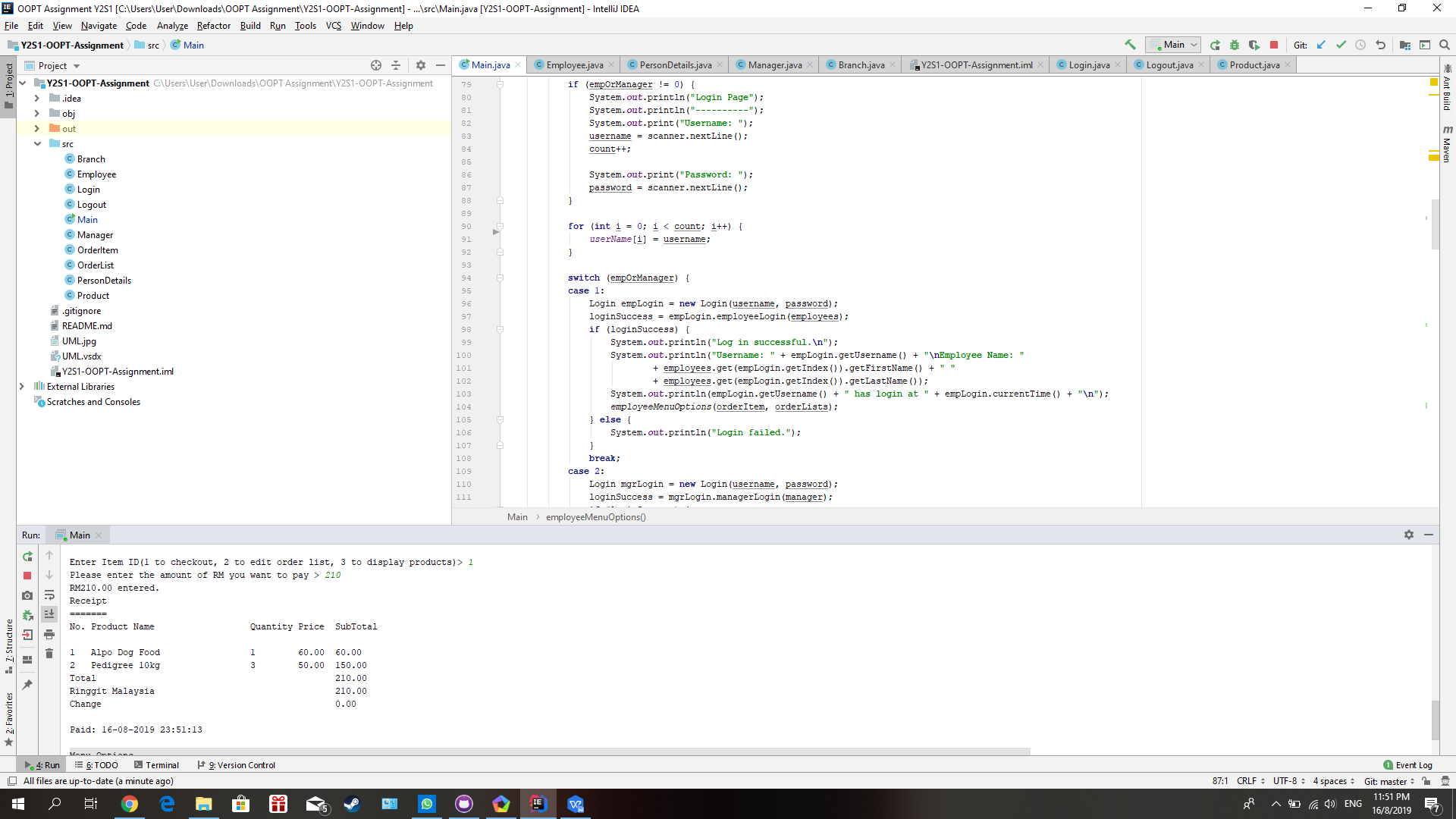


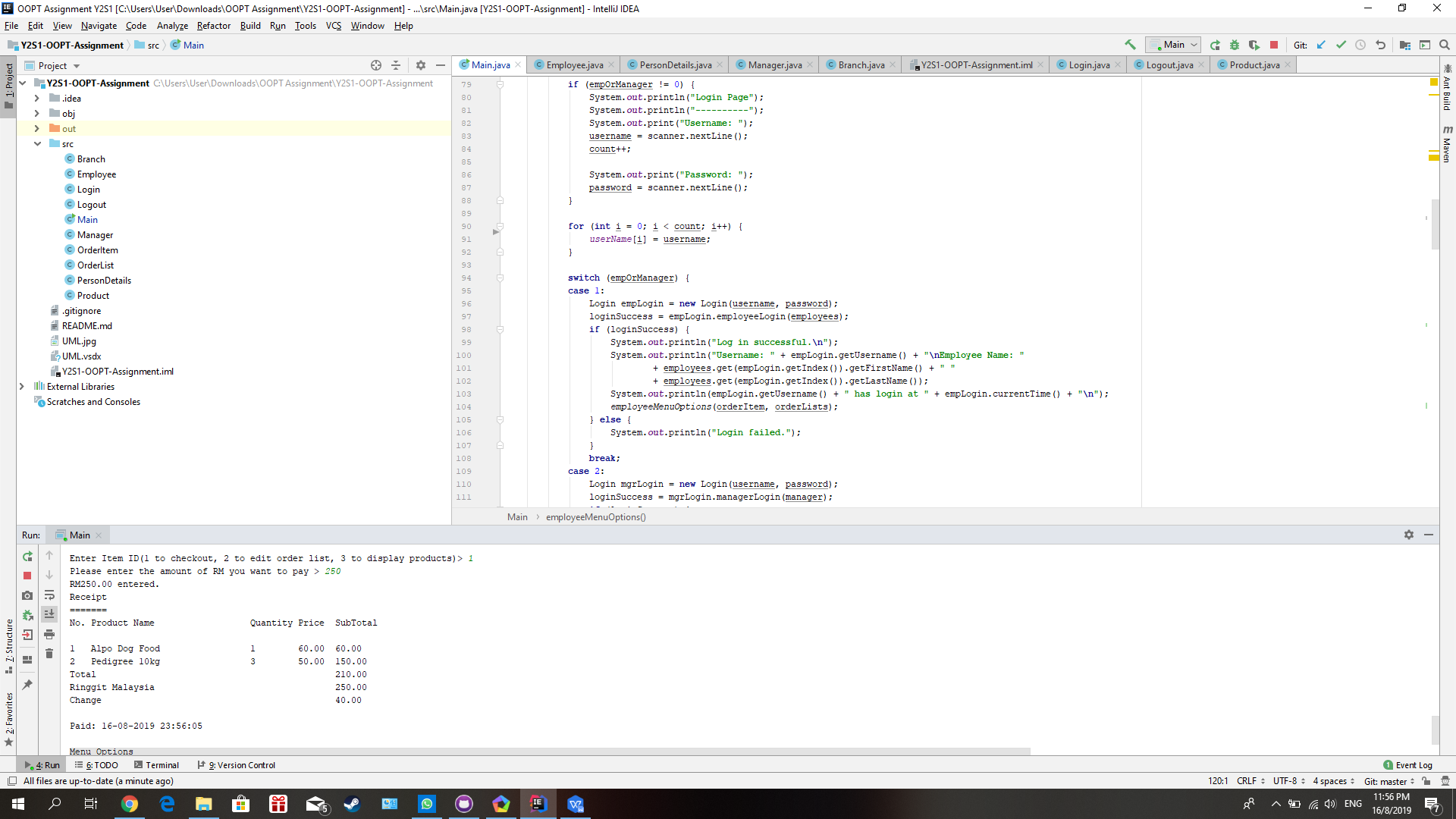


User can modify the quantity of item ordered by entering the number 2, then it will prompt user to enter the number of quantity the user want to modify.



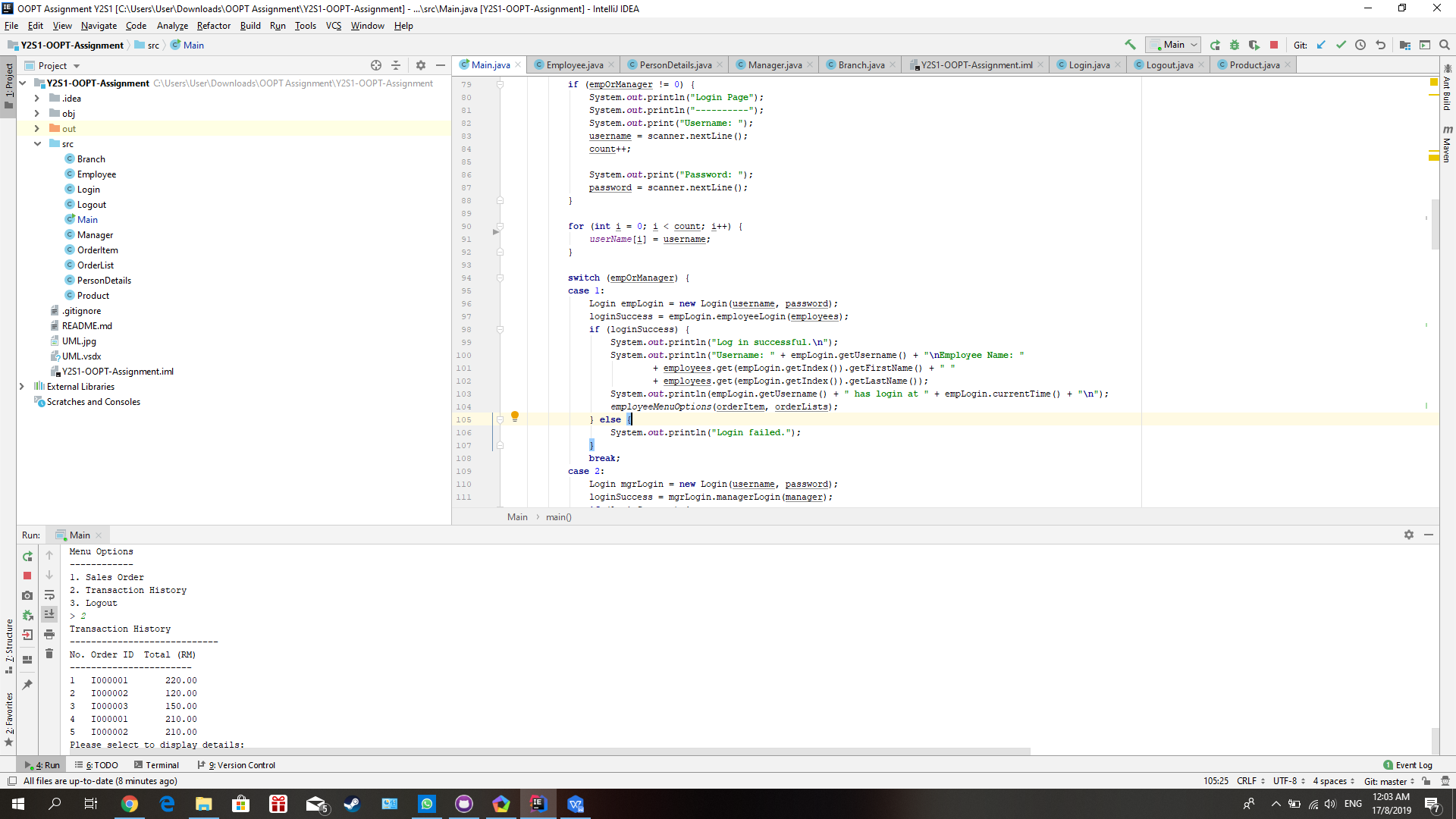
User can enter the number 1 to checkout, If the user enter the amount of RM to pay is less then the total amount ordered, it will display an error message to the user such as the picture shown.



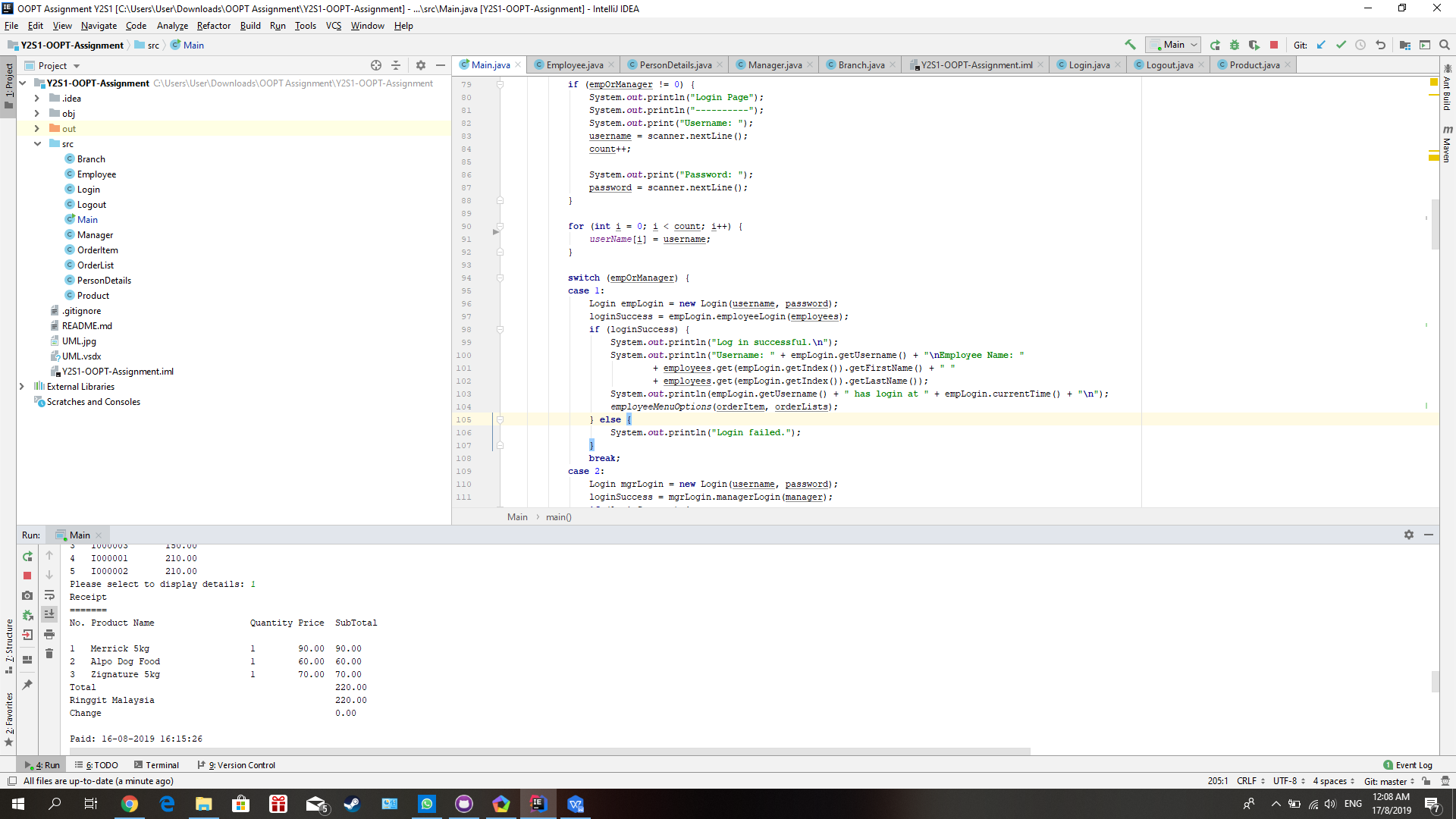


If the user enter the amount of RM to pay is equal to or more than the total amount ordered, it will generate and display the receipt and will have change if the amount of customer paid is more than the total amount ordered such as two of the pictures shown. It will also display the date and time when the receipt is generated

1. **Transaction History**

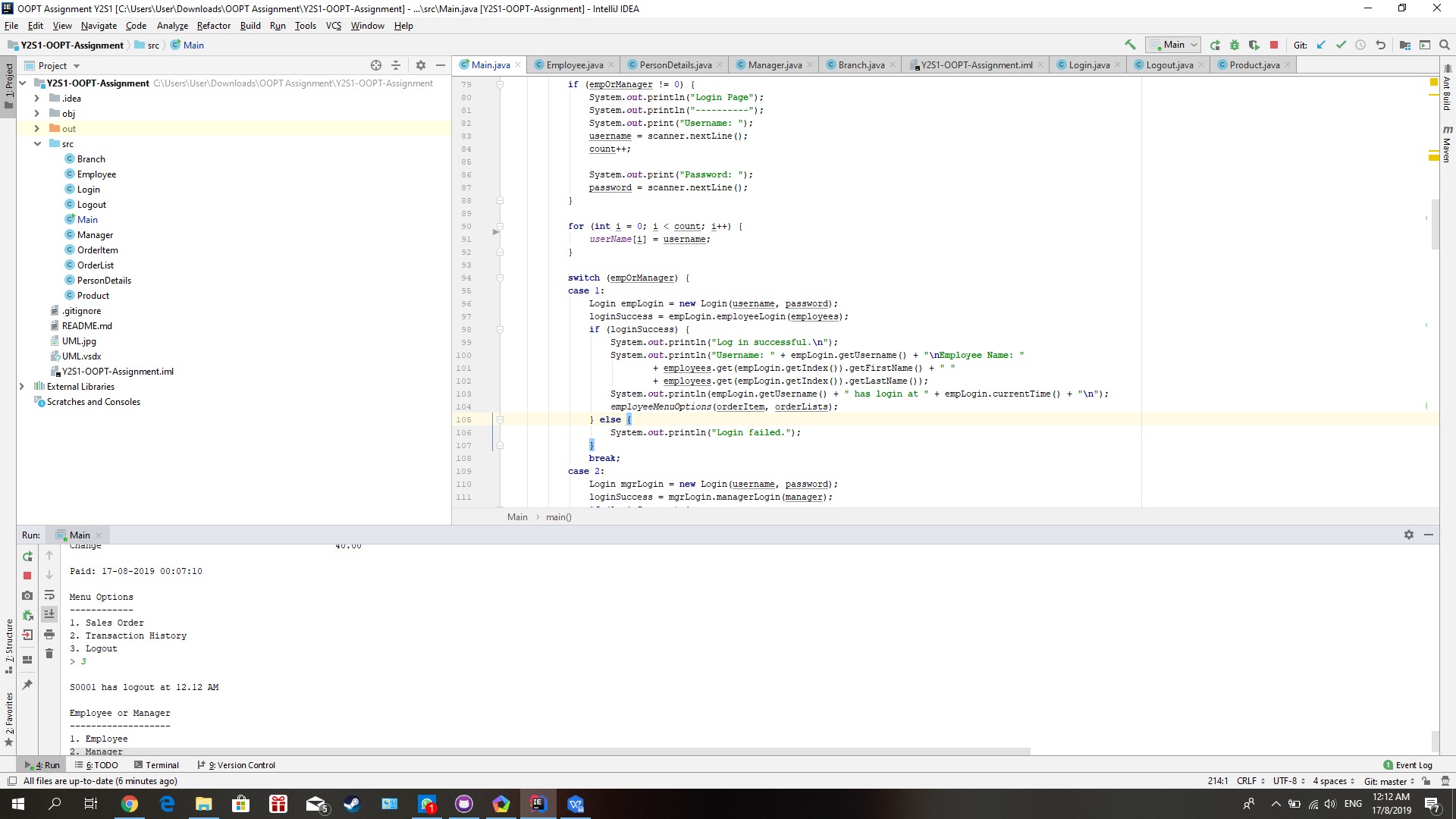


User can enter the number 2 in the menu options to check the transaction history.



User can check the transaction history by entering valid input such as the number of 1 to 5 as the picture shown. It will display the transaction history to the user based on the number of the user enter.

1. **Logout**



User can logout by entering the number of 3 in the menu options. When the user has logged out, it will display the time of user logout.