**[ Day 2]**

**Class, Object creation and basic method operations:**

Create a class Customer with following private variables

custNo int

custName String

custAddr String

* Define init() method for initializing all variables to some default values.
  + E.g. custName=”Ram”
* Define display() method for displaying all the variables.

Create a class PizzaHut having main method and do the following

* Create an object of Customer
* Use init() for initializing an object of Customer
* Use display() for displaying details of Customer.

Setters and getters:

Modify above assignment and define setters and getters methods to initialize or set individual

variables. Call them from main methods of PizzaHut class.

**Array operations:**

Create a class Customer with following private variables

custNo int

custName String

custAddr String

Define default and parameterized constructor to initialize all variables.

Create a class CustomerReport with following variable

custList Customer[] array of type customer

Define following methods in it

addCustomer(Customer) to add a customer into array

printList() to iterate through the array and print details of each Customer

Create a class PizzaHut having main method and do the following

1. Accept values from user for creating 5 Customer objects by using constructors.
2. Store them into an array of type Customer of CustomerReport class .
3. Use addCustomer() to add each of customer to the array.
4. At last print report of customer by using printList().

**Static variables, methods and static initialization block:**

Modify Customer class and do the following

1. Define a private static variable billNo.
2. Initialize the static variable in static initialization block.
3. Bill no should be automatically incremented per customer object is created.
4. Define static method printBillNo() to print bill no

Modify main method to Print the output in following format

Bill No :\_\_\_\_\_\_\_\_ Date : \_\_\_\_\_\_\_\_\_\_

Customer : \_\_\_\_\_\_\_\_\_\_\_\_\_

Address : \_\_\_\_\_\_\_\_\_\_\_\_\_

**Inheritance and method overriding**

Create a class Pizza with following variables and methods

type String

toppings String [E.g. Mushroom, olives etc. Use No in case no toppings]

name String

timeForPreparation float

* Type can be either veg or non-veg. Do the validation in constructor. Display the message “Invalid type” in case validation fails.
* Define toString() method to display details of pizza

Create a class ItalianPizza inheriting Pizza.

* Override toString() to print E.g.

“Italian Pizza details are: Veg Margherita Pizza with Mushroom toppings which take 10

min for preparation. ” (values of variables are placed in between sentence.)

Create a class MexicanPizza inheriting Pizza.

* Override toString() to print E.g.

“Mexican Pizza details are : Non-Veg Mexican Pizza with chicken toppings which takes 15 min for preparation. ” (values of variables are placed in between sentence.)

Create a class PizzaHut having main method and achieve following functionality.

1. Ask user to choose option from Italian or Mexican pizza
2. Based upon user selection create an object of appropriate class by accepting values from user.
3. Print the details of pizza.
4. Repeat first 3 steps as many times user wants.

**[ Day 3 ]**

**Abstract classes, packages**

Create an abstract class Pizza with following private variables

type String

toppings String

name String

timeForPreparation float

costOfPizza int

size String

* + Create a parameterized constructor to accept all variables and do the appropriate initialization.
  + Type of pizza can be either “Veg” or “Non-Veg”. If validation fails show message “Only Veg and Non-Veg type is allowed”.
  + Size of pizza can be either “small” or “medium. If validation fails show message “Only small and medium size pizzas are available”.
  + Do all the validation in constructor
  + Define an abstract method int calculateCost() to calculate total cost of pizza and return it.

Create a class ItalianPizza inheriting Pizza

* Override calculateCost(). Cost of ItalianPizza Will be calculated as

|  |  |
| --- | --- |
| Veg Pizza small size | 200 |
| Veg Pizza medium size | 350 |

|  |  |
| --- | --- |
| Non-Veg Pizza small size | 270 |
| Non-Veg Pizza medium size | 420 |

|  |  |
| --- | --- |
| Added toppings | 30 |

* Based upon above conditions calculate total cost of pizza.
* Override the toString() method to print all details of Pizza along with its total cost

Create a class Delivery having main method and achieve following functionality in it.

1. Accept input from user and create an object of Italian pizza
2. Give user choices eg. Veg or Non-Veg

Small or Medium

Added Toppings

1. Based upon users selection calculate total cost of pizza and display it on console.

**Interfaces**

Create an interface Deliverable in com.pune.pizzahut package having following data

deliveryAreaLimit int (Eg. 2 km where home delivery is available)

Define a method boolean delivery()

Create a class Order implementing Deliverable with following private variables and methods

orderNo int

orderDate Date

cost int

custName String

custAddress String

approxDistance int

* Define parameterized constructor to initialize all variables.
* Override delivery() to define the delivery mode. Method returns true if approx. distance matches with the deliveryAreaLimit. Else display a message “Home Delivery not available for this distance.”

Create a class Reception having main method and perform following functionality in it

1. Accept values from user to create an order object with appropriate values
2. Reply to the user request by calling delivery() method. i.e. whether order is deliverable or not.

**[ Day 4 ]**

**Exception Handling**

Create a class InvalidPizzaTypeException and InvalidPizzaSizeException which will be user defined exception.

Modify Pizza class created in previous day assignment as bellow

* In case validation for type fails in constructor, throw a user defined checked exception called

InvalidPizzaTypeException.

* In case validation for size of pizza fails, throw a user defined checked exception called

InvalidPizzaSizeException.

* Handle these exceptions in main.

Create a class Bill with following private variables and methods

billNo int

custName String

date Date

total int

items String[ ]

Create getters and setters for all of the variables.

Create a class BillReports with following private variables and public methods

Bill bills[] i.e. array of type bill

Define methods as follows

void addBill(Bill) method to add bill object into an array

boolean updateBill(billNo,newTotal) accepts billNo and modifies the total of particular bill,

returns true if successfully updated

Bill[ ] getBills() return an array of type Bill

Bill searchBill(billNo) accepts bill no and returns details of Bill.

* In searching, in case Bill is not found in array throw user defined exception BillNotFoundException. Handle this exception in main.

Create a class Reports having main method with following functionality

1. Accept values from user and create object of Bill.
2. Store objects into array of BillReports class by using addBill() method.
3. Print details of an array by using getBills() method.
4. Accept billNo from the user along with modified bill amount. Find corresponding bill in array and update the amount. Print updated details. Use updateBill() for achieving this.
5. Accept billNo from the user and print details of bill. Use serachBill() method for achieving this.

**[ Day 5 ]**

**Multithreading**

Create a class Pizza with following variables and methods

type String [ Veg / Non-Veg ]

toppings String

name String

timeForPreparation float

size String [Small / Medium]

* Throw InvalidPizzaTypeException and InvalidPizzaSizeException when validation of type and size fails.
* Define toString() method to display details of pizza

Create a Thread class CookThread with following private variables

name String

pizza Pizza

Create a Thread class WaiterThread with following private variables

name String

pizza Pizza

Create a class PizzaHut with main method and implement following functionality into it

1. Create an object of Pizza by accepting input from the user
2. Start CookThread to cook the pizza as per user specifications
3. Start WaiterThread to serve the pizza as soon as CookThread creates it
4. Both Thread should work in synchronize manner to serve pizzas to customer .i.e. CookThread should create a Pizza and WaiterThread should serve it immediately.

[Hint : Use wait, notify methods along with synchronization to do this.]

**[Day 6]**

**Collections**

Create a class Bill with following private variables and methods

billNo int

custName String

date Date

total int

pizza Vector

Create getters and setters for all of the variables.

Create a class Reception having main method and perform following functionality

* Accept input from user to create a Bill
* Each user can order multiple types of Pizzas and their information will be stored in Vector of type pizza. Use hierarchy of pizza created in day 2 (inheritance) assignments.

[ Day 7 ]

**FileHandling-Serialization**

Create a class History to save history of bills. Define following private variables and methods

bills ArrayList

Create getters setters for variable.

Define methods as per follow guidelines

saveHistory() to save bills in a file

retriveBills() to retrieve bills from file

Create a class Reception with main method and achieve following functionality in it

1. Display following menu on console
   1. Create bills
   2. Save to File
   3. Retrieve from file
   4. Exit
2. When selected “Create bills”, it should prompt the user to accept 5 inputs as mentioned.
3. When selected “Save to File”, it should call saveHistory() to save ArrayList to “bills.dat”
4. When selected “Retrieve from File”,it should call retriveBills() to retrieve bills and display on the console.
5. Exit should exit from the application

**File Reading and Writing**

Create a class Customer with following private variables

custNo int

custName String

custAddr String

Define getters and setters for all the variables.

Create a class CustomerReport with following method

void storeReport(Customer) to store Customer details into file

Create a class Reception having main method and achieve following functionality into it

1. Accept values from user to create Customer object
2. Call storeReport() of CustomerReport to accept the customer object and store details into a file “CustomerRecords.txt” in following format.

Customer No Customer Name Customer Address

1211 Ram Singh Hinjewadi,Pune

1222 Sharda Roy Mumbai