LaboratoryExperiments

# Name:Aditya shinde

# PRN:2124UMLM1110

**Q1.BasicJavaProgramwithInput/Output**

**Aim:**TowriteaJavaprogramthatacceptsstudentdetails(name,rollnumber,andmarks)usingconsole inputanddisplaysthedetailsinaformattedoutput.

**Theory:**InputandoutputarethefoundationofuserinteractioninJava.

**Input:**TheScannerclass(fromjava.utilpackage)iscommonlyusedforreadinginputfromthe keyboard.

**Output:**JavausesSystem.out.print()andSystem.out.println()fordisplayingdata.

**FormattedOutput:**Theprintf()methodprovidesbettercontrolovertheappearanceofnumbersand text.ThisprogramdemonstrateshowJavareadsanddisplaysdataofdifferenttypesandprintsthem inanorganizedmanner.

Code:

importjava.util.Scanner;

classStudentDetails{

publicstaticvoidmain(String[]args){ Scannersc=newScanner(System.in);

System.out.print("EnterName:"); Stringname=sc.nextLine();

System.out.print("EnterRollNo:"); introllNo=sc.nextInt();

System.out.print("EnterMarks:"); floatmarks=sc.nextFloat();

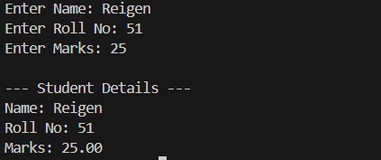
System.out.println("\n--- StudentDetails---");

System.out.printf("Name:%s\nRollNo:%d\nMarks:%.2f\n",name,rollNo,marks);

}

}

Output:



# Q2.Decision-MakingSystemforLibrary

**Aim:**TodevelopaJavaprogramformanaginglibrarybookcategoriesusingconditionalstatementsto check book availability.

**Theory:**Decision-makingstatementsallowaprogramtochooseactionsbasedonconditions.

The**if-else**statementexecutesablockofcodeonlyifacertainconditionistrue.

The**switch**statementisusedforselectingoneoptionfrommultiplechoices.Thisconceptisuseful inapplicationslikelibrarysystems,wherethesystemmustdecidewhichmessageordatatodisplay based on user input.

Code:

importjava.util.Scanner;

classLibrarySystem{

publicstaticvoidmain(String[]args){ Scannersc=newScanner(System.in);

System.out.println("Entercategory(Fiction/Science/History):");

Stringcategory=sc.nextLine();

if(category.equalsIgnoreCase("Fiction")) System.out.println("Fictionbooksavailable:10");

elseif(category.equalsIgnoreCase("Science"))

System.out.println("Sciencebooksavailable:5"); elseif(category.equalsIgnoreCase("History"))

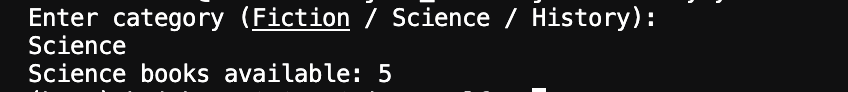
System.out.println("Historybooksavailable:8"); else

System.out.println("Invalidcategory.");

}

}

Output:



# Q3.FlightReservationSystemusingLoops

**Aim:**Todesignareservationsystemusingloopsthatallowsuserstobookorcancelflightticketsmultiple timesuntiltheyexit.

**Theory:**Loopsenablerepetitiveexecutionofcodeblocks.

The**whileloop**runsuntilaconditionbecomesfalse.

The**forloop**isusedforcountingiterations.Inthisexperiment,loopsallowcontinuoususer

interaction,sotheusercanrepeatedlybookorcancelticketswithoutrestartingtheprogram.This modelsareal-worldinteractivesystem.

Code:

importjava.util.Scanner;

classFlightReservation{

publicstaticvoidmain(String[]args){

Scannersc=newScanner(System.in); int seats = 5;

while(true){

System.out.println("1.BookTicket2.CancelTicket3.Exit"); intchoice=sc.nextInt();

if(choice==1CCseats>0){ seats--;

System.out.println("Ticketbooked!Remainingseats:"+seats);

}elseif(choice==2){ seats++;

System.out.println("Ticketcancelled.Seatsavailable:"+seats);

}elseif(choice==3) break;

else

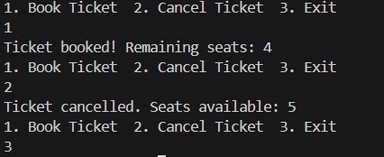
System.out.println("Invalidornoseatsavailable.");

}

}

}

Output:



# Q4.Array-BasedPassengerBookingSystem

**Aim:**Toimplementa2Darraythatstoresanddisplaysbookingdetailssuchaspassengername,seat number, and class.

**Theory:**Arraysareusedtostoremultiplevaluesofthesametypeincontiguousmemorylocations.A**2D array**canbeimaginedasatable(rowsandcolumns)thatholdsstructureddatalikeapassenger list.Usingloopswitharraysallowssystematicdatastorageandretrieval,whichisvitalinbookingand reservationsystems.

Code:

importjava.util.Scanner;

classPassengerBooking{

publicstaticvoidmain(String[]args){ Scannersc=newScanner(System.in);

String[][]bookings=newString[3][3];//name,seatno,class

for(inti=0;i<3;i++){

System.out.print("EnterName:"); bookings[i][0]=sc.next(); System.out.print("EnterSeatNo:"); bookings[i][1]=sc.next(); System.out.print("EnterClass:"); bookings[i][2]=sc.next();

}

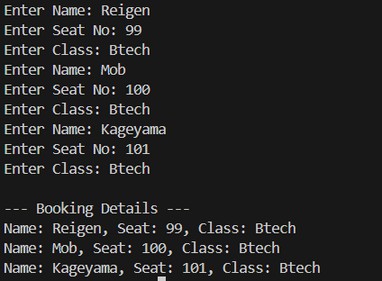
System.out.println("\n--- Booking Details ---"); for(String[]b:bookings)

System.out.println("Name:"+b[0]+",Seat:"+b[1]+",Class:"+b[2]);

}

}

Output:



# Q5.BankingTransactionTrackerusingStrings

**Aim:**TodevelopaprogramthatrecordsbankingtransactionsusingStringandStringBuilder,allowingthe usertoappendandviewthetransactionhistory.

**Theory:**StringsinJavaareimmutable(cannotbechangedoncecreated),soforcontinuoustext modification,**StringBuilder**ispreferred.Itallowsoperationslikeappend(),insert(),anddelete()

efficiently.Thisexperimentdemonstratesdynamicstringmanipulationtomaintainatransactionlog—an

importantconceptinapplicationslikedigitalbankingsystems.

Code:

importjava.util.Scanner;

classBankingTransaction{

publicstaticvoidmain(String[]args){ Scannersc=newScanner(System.in);

StringBuilderhistory=newStringBuilder();

while(true){

System.out.println("1.Deposit2.Withdraw3.ShowHistory4.Exit"); intch=sc.nextInt();

if(ch==1){

System.out.print("Amount:"); intamt=sc.nextInt();

history.append("Deposited:").append(amt).append("\n");

}elseif(ch==2){

System.out.print("Amount:"); intamt=sc.nextInt();

history.append("Withdrawn:").append(amt).append("\n");

}elseif(ch==3) System.out.println(history.toString());

else

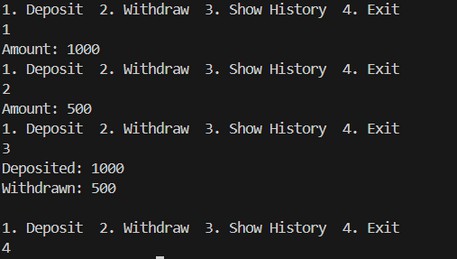
break;

}

}

}

Output:



# Q6.FileI/OforCustomerAccounts

**Aim:**ToreadandwritecustomeraccountdetailsintoatextfileusingfilehandlinginJava.

**Theory:**Filehandlingenablesdatatobepermanentlystoredandretrieved.Javaprovidesclassessuchas:

**FileWriter** and **BufferedWriter** for writing data into files.

**FileReader**and**BufferedReader**forreadingdatafromfiles.Thisconceptiscrucialfordeveloping systemsthatstoredatapersistently,likebankingsystems,reservationapps,andemployee

databases.

Code:

importjava.io.\*;

importjava.util.Scanner;

classCustomerFileIO{

publicstaticvoidmain(String[]args)throwsException{ Scanner sc = new Scanner(System.in);

System.out.print("EnterName:");

Stringname=sc.nextLine(); System.out.print("EnterBalance:"); doublebal=sc.nextDouble();

FileWriterfw=newFileWriter("customer.txt"); fw.write("Name:"+name+"\nBalance:"+bal); fw.close();

BufferedReaderbr=newBufferedReader(newFileReader("customer.txt")); Stringline;

System.out.println("\nFileContent:");

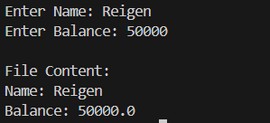
while((line=br.readLine())!=null) System.out.println(line);

br.close();

}

}

Output:



# Q7.ShoppingCartwithPriceFormatting

**Aim:**Tobuildashoppingcartsystemthatcalculatestotalcost,GST,anddisplaysfinalpriceusingproper numericformatting.

**Theory:**Pricecalculationofteninvolvesfloating-pointarithmeticandrequiresformattedoutput.The

**DecimalFormat**class(fromjava.textpackage)helpsformatnumericalvalues(e.g.,twodecimal points).Thisexperimentsimulatesane-commercecart,whereusersinputproductprices,andthe systemautomaticallycalculatestotalpriceandapplicabletax(GST).

Code:

importjava.text.DecimalFormat; importjava.util.Scanner;

classShoppingCart{

publicstaticvoidmain(String[]args){ Scannersc=newScanner(System.in);

DecimalFormatdf=newDecimalFormat("#.00"); double total = 0;

for(inti=1;i<=3;i++){

System.out.print("Enterpriceofitem"+i+":"); total+=sc.nextDouble();

}

doublegst=total\*0.18;

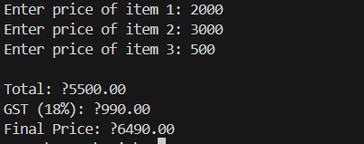
doublefinalPrice=total+gst;

System.out.println("\nTotal:₹"+df.format(total)); System.out.println("GST(18%):₹"+df.format(gst)); System.out.println("Final Price: ₹" + df.format(finalPrice));

}

}

Output:



# Q8.DateandPatternMatchingforOrders

**Aim:**TowriteaprogramthatextractsspecificdetailslikeOrderIDfromtextusingRegularExpressions (Regex)anddisplaysthecurrentdate.

**Theory:**Regularexpressions(Regex)arepatternsusedfortextsearchingandvalidation.InJava:

**Pattern**classdefinestheregularexpression.

**Matcher**classsearchesforpatternswithinstrings.Combinedwith**datehandling**usingthejava.time package,thisexperimenthelpsinextractingstructuredinformationfromunstructureddata,useful ininvoiceorordermanagementsystems.

Code:

importjava.util.regex.\*; importjava.time.\*;

classOrderPattern{

publicstaticvoidmain(String[]args){

Stringorder="Customer:Riya,OrderID:12345,Date:2025-10-13"; Patternp=Pattern.compile("OrderID:(\\d+)");

Matcherm=p.matcher(order);

if(m.find())System.out.println("OrderID:"+m.group(1));

LocalDatetoday=LocalDate.now();

System.out.println("CurrentDate:"+today);

}

}

Output:



# Q9.GUIApplicationwithSwingforPersonalAssistant

**Aim:**Tocreateasimplegraphicaluserinterface(GUI)usingJavaSwingthatallowsuserstoenterandsave tasks.

**Theory:**SwingisapartofJava’sjavax.swingpackageandisusedtobuildwindow-basedapplications.Key componentsinclude:

**JFrame**: The main window.

**JLabel**: Displays text.

**JTextField**: Allows user input.

**JButton**:Performsanactionwhenclicked.Thisexperimentintroducesevent-drivenprogramming, wheretheprogramreactstousereventssuchasbuttonclicks.

Code:

importjavax.swing.\*;

classPersonalAssistant{

publicstaticvoidmain(String[]args){

JFrameframe=newJFrame("PersonalAssistant"); JLabellabel=newJLabel("EnterTask:");

JTextFieldfield=newJTextField(15); JButtonbtn=newJButton("Save");

JPanelpanel=newJPanel();

panel.add(label); panel.add(field); panel.add(btn);

frame.add(panel);

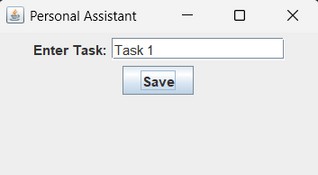
frame.setSize(300,150);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setVisible(true);

}

}

Output:



# Q10.Event-BasedExceptionHandlinginPersonalAssistantApp

**Aim:**TodevelopaGUIapplicationthatdemonstrateseventhandlingandexceptionhandlingforinvalid inputs.

# Theory:

**EventHandling:**JavausesinterfaceslikeActionListenertoperformactionswhenanevent(likea button click) occurs.

**ExceptionHandling:**Thetry-catchmechanismhandlesruntimeerrorsgracefullywithoutcrashing theprogram.Thisexperimentcombinesuserinterfaceanderrorhandling,teachinghowtomake applicationsbothinteractiveandrobust.

Code:

importjavax.swing.\*;

importjava.awt.event.\*;

classAssistantException{

publicstaticvoidmain(String[]args){

JFramef=newJFrame("EventHandler"); JTextFieldfield=newJTextField(10); JButtonbtn=newJButton("Check");

btn.addActionListener(e->{ try{

intnum=Integer.parseInt(field.getText());

JOptionPane.showMessageDialog(f,"Number:"+num);

}catch(NumberFormatExceptionex){

JOptionPane.showMessageDialog(f,"Invalidinput!");

}

});

JPanelp=newJPanel(); p.add(field);

p.add(btn);

f.add(p); f.setSize(250,120);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setVisible(true);

}

}

Output:

