**INTRODUCTION**

This report discusses the result of the work done in development of "BILLING MANAGEMENT SYSTEM" on Java Platform. It is a part of the project going in IT Department, and aims at the development of an application framework for providing a common platform for facilitating the use of methodological approach developed by the ASSET team and integration of various tools developed during the execution of the project.

**PROJECT AIMS AND OBJECTIVES**

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

* Billing system.
* Request column for customers to buy new product(s).
* Employee login page where employee can login by their appropriate id’s.
* Employees have to enter the product description to get the total of the product which the customer wants to buy.
* Save time

**BACKGROUND OF PROJECT**

TechnoShop Billing Systemis an application which refers to billing systems which are generally small or medium in size. It is used by employees of a shop to get the sub total of the products using a computerized system where he/she can record various transactions like buying of new product, addition of new product(s), changes in price of product(s) etc.

Product(s) maintenance modules are also included in this system which would keep track of the all the products using the and also a detailed description about the products the shop contains. With this computerized system there will be no error in calculations or record of the products present in the shop which generally happens when a non-computerized system is used.

In addition, report module is also included in Billing Management System. The user is able to generate different kinds of reports like lists of products sold, list of products.

All these modules are able to help employee to manage the shop with more convenience and in a more efficient way as compared to billing systems which are not computerized.

**OPERATION ENVIRONMENT**

|  |  |
| --- | --- |
| PROCESSOR | INTEL CORE PROCESSOR OR BETTER PERFORMANCE |
| OPERATING SYSTEM | WINDOWS VISTA, WINDOWS7, WINDOWS8 |
| MEMORY | 1GB RAM OR MORE |
| HARD DISK SPACE | MINIMUM 3 GB FOR DATABASE USAGE FOR FUTURE |

**SYSTEM ANALYSIS**

Here, we will discuss and analyse about the developing process of TechnoShop Billing System including software requirement specification (SRS) and comparison between existing and proposed system . The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing v/s proposed provides a view of how the proposed system will be more efficient than the existing one.

**SOFTWARE REQUIREMENT SPECIFICATION**

**GENERAL DESCRIPTION**

PRODUCT DESCRIPTION:TechnoShop Billing Systemis a computerized system which helps user (employees) to manage the shop daily activity in electronic format. It reduces the risk of paper work such as file lost, file damaged and time consuming. It can help user to manage the transaction or record more effectively and timesaving.

PROBLEM STATEMENT: The problem occurred before having computerized system includes:

* Bill lost- When computerized system is not implemented bill is always lost because of human environment. Sometimes due to some human error there may be a loss of records.

* Bill damaged- When a computerized system is not there bill is always lost due to some accident like spilling of water by some member on file accidentally.
* Difficult to search record- When there is no computerized system there is always a difficulty in searching of records if the records are large in number .
* Space consuming- After the number of records become large the space for physical storage of file and records also increases if no computerized system is implemented.
* Cost consuming- As there is no computerized system the to add each record paper will be needed which will increase the cost for the management of bill.

**SYSTEM OBJECTIVES**

* Improvement in control and performance- The system is developed to cope up with the current issues and problems of bills .The system can add product(s), validate product(s) and is also bug free.
* Save cost- After computerized system is implemented less human force will be required to maintain the bills thus reducing the overall cost.
* Save time employee is able to add product(s) by using few clicks of mouse and can calculate the total thus saving his valuable time.

**SYSTEM REQUIREMENTS**

NON FUNCTIONAL REQUIREMENTS

* Product requirement
* EFFICIENCY REQUIREMENT When a billing management system will be implemented employee will easily access shop and bill transaction will be very faster.
* RELIABILITY REQUIREMENT The system should accurately performs product registration, product validation, bill generation, bill transaction.
* USABILITY REQUIREMENT The system is designed for a user friendly environment so that staff of shop can perform the billing easily and in an effective way.
* ORGANIZATIONAL REQUIREMENT IMPLEMENTATION REQUIREMNTS In implementing whole system it uses JAVA in front end.
* DELIVERY REQUIREMENTS The whole system is expected to be delivered in one month of time with a weekly evaluation by the project guide.

**FUNCTIONAL REQUIREMENTS**

**USER LOGIN**

This feature used by the staff to login into system. They are required to enter user id and password before they are allowed to enter the system .The user id and password will be verified and if invalid id is there user is allowed to not enter the system.

**\*functional requirement**

-user id is provided when they register

-The system must only allow user with valid id and password to enter the system

-The system performs authorization process which decides what user level can access to.

-The user must be able to logout after they finished using system.

**REGISTER NEW PRODUCT**

This feature can be performed by all users to register new product(s).

**\*functional requirements**

-System must be able to verify information

-System must be able to delete information if information is wrong

**BUYING OF PRODUCT**

This feature allows to buy product and also view bills of product sold.

**\*functional requirements**

-System must be able to enter issue information in database.

-System must be able to update number of product(s).

-System should be able to enter price information.

**SOFTWARE AND HARDWARE REQUIREMENT**

This section describes the software and hardware requirements of the system.

**SOFTWARE REQUIREMENTS**

* Operating system- Windows 7 is used as the operating system as it is stable and supports more features and is more users friendly. It can be 32 bit or 64 bit.
* Operating system which supports JDK (java development kit). Download jdk as per your requirements.
* Development tools and Programming language- JAVA is used to write the whole code and develop with AWT and java swing functions for styling work and command prompt to compile the code and to run the whole code.

**HARDWARE REQUIREMENTS**

* Intel core i3 6nd generation is used as a processor because it is fast than other processors an provide reliable and stable and we can run our PC(personal computer) for long time. By using this processor we can keep on developing our project without any worries.
* Ram 1 GB is used as it will provide fast reading and writing capabilities and will in turn support in processing.

**EXISTING V/S PROPOSED SYSTEM**

1. Existing system does not have any facility of employee login or staff login whereas proposed system will have a facility of staff login.
2. Existing system does not have a facility of computerized bill of product(s) whereas proposed system has a facility of computerized bill of product(s).
3. Existing system does not have any facility of computerized billing.
4. Existing system does not has any option of availability of product(s) whereas proposed system will have this facility.
5. Existing system does not have any facility to generate employee reports as well buying product reports whereas proposed system provides manager with a tool to generate reports.
6. Existing system does not has any facility for any suggestions where as in proposed system after logging in to their accounts employees can know the availability of products.

**SOFTWARE** **TOOLS** **USED**

**JAVA:**

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called java byte code, instead of directly to architecture-specific machine code. Java byte code instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware.

* End users commonly use a Java Runtime environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets.

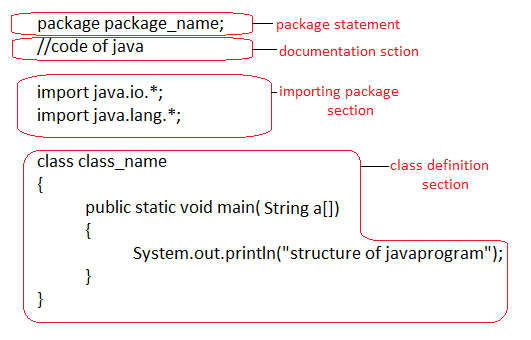
The use of universal byte code makes porting simple. However, the overhead of interpreting byte code into machine instructions made interpreted programs almost always run more slowly than native executable. Just-in-time (JIT) compilers that compile byte codes to machine code during runtime were introduced from an early stage. Java itself is platform-independent, and is adapted to the particular platform it is to run on by a Java virtual machine for it, which translates the Java byte code into the platform's machine language.

**JAVA SWING APPLICATION:**

**Swing** is a GUI widget toolkit for Java. It is part of Oracle’s Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

* Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT.
* In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

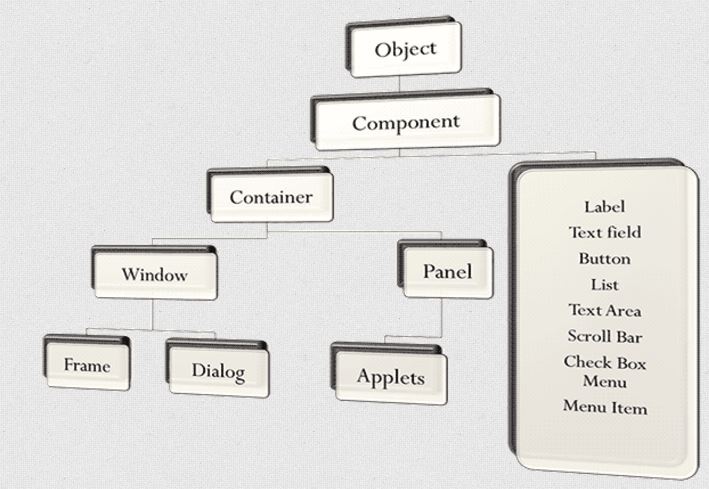
Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.



**AWT (ABSTRACT WINDOW TOOLKIT):**

Java programming language provides with classes and interface for creating and running a GRAPHICAL USER INTERFACE application. The first set of classes which were provided by java programing language for creating graphical user interface applications were part of a package called AWT (ABSTRACT WINDOW TOOLKIT).

* AWT (ABSTRACT WINDOW TOOLKIT) is an API package provided by java class library for creating and running GUI application. This API package provides pre-written classes for various types of control and either event handling.
* AWT (ABSTRACT WINDOW TOOLKIT) provides us dedicated classes for each control we need to create object of those classes if we wish to use controls provided by those classes in our application.



Various types of controls in JAVA PROGRAMMING LANGUAGE are as follows:

* **Frame**: A Frame is a top-level window with a title and a border. A frame, implemented as an instance of the frame class, is a window that has decorations such as a border, a title, and supports button components that close or iconify the window. Applications with a GUI usually include at least one frame. Applets sometimes use frames, as well.

Frame f=new Frame();

* **Panel**: Panel is the simplest container class. A panel provides space in which an application can attach any other component, including other panels.

The default layout manager for a panel is the FlowLayout layout manager

Panel p=new Panel();

* **Button**: Button is the most common AWT control that generates an event when it is pressed. In order to handle the event we have to implement ActionListener interface which defines actionPerformed***()*** method when a button is clicked. These are the commonly used methods with button class:

Button b=new Button("Next");

* **Label**: Label is one of the AWT control which is used for showing text to the user. It contains a string, which it displays. Labels do not have any interaction with the user. The two main methods to get and set the value of a label are as follows:

Label l = new Label();

* **RadioButton**:Radio buttons are groups of buttons in which, by convention, only one button at a time can be selected. The Swing release supports radio buttons with the JRadioButton and ButtonGroup classes. To put a radiobutton in a menu, use the JRadioButtonMenuItem class.

RadioButton jb = new RadioButton();

* **TextField**: It is a control that allows a user to enter strings, values etc in a single line text area and edit it in the same scope of single line. TextField allow us to edit the text by moving the arrow keys and performing cut, copy, paste operations on the text.

TextField tf = new TextField(15);

* **Choice**: Choice is replacement to radio buttons. When radio buttons are many to display, they flock all the frame. As an alternative, the **java.awt** package comes with **Choice**. That is, Choice is meant for **single selection** only.
* **CheckBox**: Checkbox consists of a small box that can either contain checked mark or not. It is used to turn an option on or off by returning true or false state. We can check multiple checkboxes at a same time. In order to handle the events on checkbox we have to implement ItemListener interface which defines itemStateChanged() method when a checkbox is checked.
* **List**: List interface is a subtype of the java.util.Collection interface. It represents an ordered list of objects, meaning you can access the elements of a List in a specific order, and by an index too. You can also add the same element more than once to a List .
* **TextArea**:TextArea is a control that allows editing of multiple lines of text.

TextArea ta=new TextArea ();

**Special classes and functions used in application**

* **setBackground ()**: public void setBackground (Color c ) sets the background color of the this component. Parameter: color – the color to become this component’s color. If this parameter is null then this component will inherit the background color of its parent.

p.setBackground(Color.red);

**OR**

p.setBackground(new Color(77,180,180));

* **setForeground ()**: setForeground is used to set the foreground colour i.e the colour in which text is shown.

p.setForeground (Color.red);

* **JComboBox()**: A JComboBox which lets the user choose one of several choices, can have two very different forms. The default form is the uneditable combo box, which features a button and a drop-down list of values. The second form, called the editable combo box, features a text field with a small button abutting it. The user can type a value in the text field or click the button to display a drop-down list.

String [] choice= {“A”,”B”,”C”,”D”};

JComboBox cb=new JComboBox (choice);

* **Font:** Applications that require specific fonts can bundle them and instantiate them using the createFont method. Logical fonts are the five font families defined by the Javaplatform which must be supported by any Java runtime environment: Serif, SansSerif, Monospaced, Dialog, and DialogInput.

Font font=new Font ("SansSerif", Font.BOLD, 20);

label.setFont(font);

* **ImageIcon():** An icon is an object that adheres to the Icon interface. **Swing** provides a particularly useful implementation of the Icon interface: ImageIcon , which paints an icon from a GIF, JPEG, or PNG image.

JLabel logo=new JLabel(new ImageIcon("C:\\hp\\Desktop\\ts.jpg"));

* **setPreferredSize():**  It sets the preferred size. The Layout Manager will try to arrange that much space for your component. It depends on whether you're using a layout manager or not.

Button pro=new Button ("PROCEED");

pro.setPreferredSize(new Dimension(100,40));

* **setVisible ()**: It controls whether a component and its children are displayed on the screen or not.

f.setVisible(true);

* **setLocation ()**: setLocation() method to specify the size and position of a widget or component. It is used to positioning of label, button and other widgets.

f.setLocation(250,160);

* **setTitle()**: setTitle method to change the variable in the PointGraphWriter class. Therefore, you are trying to set the title of the frame to be a null  String because the setTitle method is only called after the constructor method.

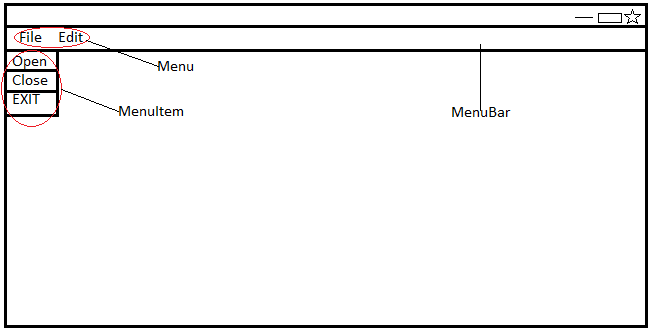
f.setTitle("TechnoShop BILLING SYSTEM");

* **setResizable():** one method for ensuring that a graphical interface looks the way you intend is to prevent the user from re-sizing it. If the parameter is false then the user cannot re-size the frame.

f.setResizable(false);

**MENU BASED APPLICATION**

Java.awt package provides us with pre-defined classes for creating a menu based GUI application. We need to instantiate these classes to provide a menu bar in a GUI application.



The following 3 classes are used for creating a menu based GUI application.

* MenuBar
* Menu
* MenuItem

**CONTROL STATEMENTS IN JAVA:**

Java statement is the smallest unit that is complete instruction in itself. Statements in java generally contain expressions and end with a semi-colon. The two most commonly used statements in any programming language are as follows:

* **Sequential statements:** These are the statements which are executed one by one.
* **Control statements:** These are the statements that are executed randomly and repeatedly. These are followed by Java also.

These statements are executed by JVM one by one in a sequential manner. So they are called sequential statements. But this type of sequential execution is useful only to write simple programs. If we want to write better and complex programs, we need better control on the flow of execution. This is possible by using control statements.

Control statements are the statements which alter the flow of execution and provide better control to the programmer on the flow of execution. They are useful to write better and complex programs.

The following control statements are available in Java:

* if ...else statement
* do...while loop
* while loop
* for loop
* for-each loop
* switch statement
* break statement
* continue statement

**if...else statement:** This statement is used to perform a task depending on whether a given condition is true or false. The condition is written inside the small braces (). The statements written inside the square brackets [ ] represent optional part of the statement. It means that the part within [ ] can be omitted, if not required. This is the convention followed in all control statements.

if(me.getSource()==pro)

{

p1.setVisible(false);

scr2.setVisible(true);

f.add(scr2);

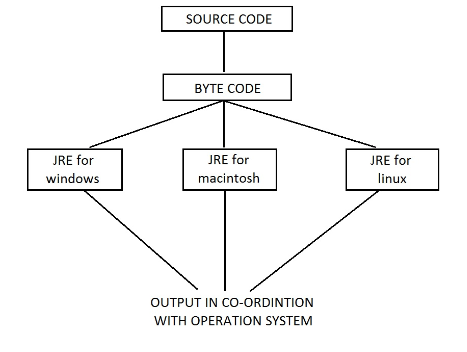
f.setVisible(true);

}

**For loop:** The for loop is also same as do...while or while loop, but it is more compact syntactically. The for loop executes a group of statements as long as· a condition is true.

**Stages in Execution and Compilation of a JAVA Language Program**

A java source code is saved in a file with (.java) extension. Java source code is compiled using a command line tool called javac (java compiler). This tool generates byte code which is an highly optimised set of instruction. This byte code is purely based on source code so is architecture neutral.



* This byte code is interpreted by a tool called JVM (java virtual machine). JVM is an interpreter for the byte code
* JDK (java development kit) consist of slash provide a run time environment for the execution of a java program. This environment is called JRE (java runtime environment) there are several version of JRE. This byte code is produced by javac taken to any machine containing JRE. JVM, part of JRE, will interpret the byte code and will produce the output in co-ordination with the operating system. Since the byte code produced on any machine (architecture) can be interpreted by JRE, platform independence/ demonstrated.
* The operating system get involved when the byte code has reached the target machine.

**CONSTRUCTORS**

A constructor is a special member function which is responsible for creating object(s) in an application. A constructor is a function which creates a memory block equal to the sum of the sizes of all the members, each time the code for the object creation gets executed.

A constructor is a special member function which has same as that of the class name and no return type. A constructor is a special member function because unlike normal method(s), it gets executed automatically.

Types of constructors:

* DEFAULT constructor
* PARAMETERIZED constructor
* COPY constructor

**ARRAYS IN JAVA**

Arrays are a reference type which is used to work on data of a single data type. Array is a collection of homogenous elements/ values.

Whenever we create an array, a memory block containing elements equal to the size of array gets created in memory. Each element in the array is assigned a unique index number the index number of first element is 0 (zero), second is 1 and so on. The index number of last element is one less than the size of the array [ ].

* In java we can create only a dynamic array.

An array is a group of like-typed variables that are referenced to by common name. Array offer a convenient means if grouping related information.

An array is a single data variable that can store multiple pieces of data that are each of the same data type.

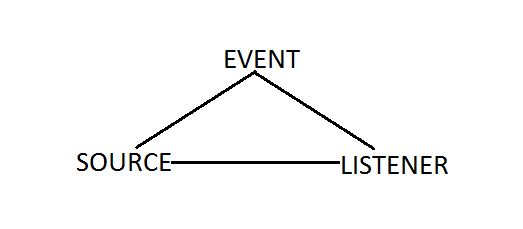
Type of arrays in java:

* SINGLE DIMNESIONAL array (1-D)
* MULTIDIMENSIONAL array (2-D, 3-D)
* JAGGED array
* The advantage of using array is that they simplify programming by replacing a lot of statement(s) by just one or two statement. Array is used to handle large amount of data without the need to declare many individual variable(s) separately.

**EVENT HANDLING**

Event handling is a special type of programming in which a given piece of given code (written in the application gets executed) when some operation is performed by the user or some event take place in the background. Event handling is the programming technique which allows us to create interactive application. An interactive app is application which is able to respond to the user input.

JAVA CLASS LIBRARY provides an API-package contains pre-defined classes and interface for enabling event handling. Java class library a specialised model called Delegation Event Model for enabling event handling.

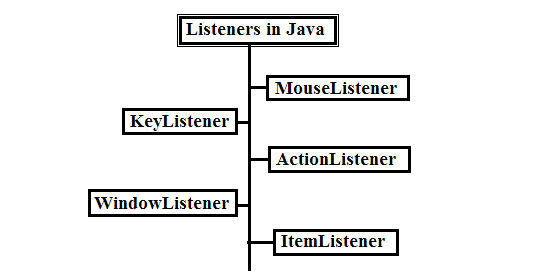


The task of event handling is performed collectively by 3 set of coding named EVENT, SOURCE, and LISTENER. SOURCE is a technical term used for the object which generated the event/ object which notices the state changed in source it finds out the nature of the operation which has performed.

EVENT is the object which notices the state change in source it finds out the nature of the operation which was performed.

LISTENER is an object which is notified when some event is performed on a source. Listener contains the coding which should be executed only and only when some operation is performed on a control.

Java class library provides dedicated listener (interfaces) for each type of event handling. Which are as follows:



**MouseListener:**

MouseListener is a public interface which is a part of java.awt.event package. This interface contains declaration of 5 methods, each pertaining to some mouse related operation. Each of the function declared in the MouseListener accepts a reference to an object of MouseEvent handling. These 5 methods are following:

Public void mousePressed(MouseEvent variable\_name);

Public void mouseReleased(MouseEvent variable\_name);

Public void mouseClicked(MouseEvent variable\_name);

Public void mouseEnetered(MouseEvent variable\_name);

Public void mouseExited(MouseEvent variable\_name);

pro.addMouseListener(this); //REGISTER

public void mouseClicked(MouseEvent me)

{

if(me.getSource()==pro)

{

p1.setVisible(false);

scr2.setVisible(true);

f.add(scr2);

f.setVisible(true);

}

**ItemListener:**

ItemListener is a public interface which is part of java.awt.event package. ItemListener is an interface listener which is used for performing event handling with controls such as radio button, check box, drop down list etc.

Item Listener contains declaration of a method named item state changed ( ) this method is called each time when an item belonging to a control which is registered which item listener is either selected or deselected.

public void itemStateChanged(itemEvent ie);

**KeyListener:**

Key Listener is used for performing event handling on text boxes and text area control.

public void keyPressed(KeyEvent variable\_name);

public void KeyReleased(KeyEvent variable\_name);

public void KeyTyped(KeyEvent variable\_name);

**ActionListener:**

It is common to have multiple ways to activate the same command. The user can choose a certain function through a menu, a keystroke, or a button on a toolbar. This is easy to achieve in the AWT event model: link all events to the same listener.

The Swing package provides a very useful mechanism to encapsulate commands and to attach them to multiple event sources: the Action listener.

The Action listener has the following methods:

void actionPerformed(ActionEvent event)

void setEnabled(boolean b) boolean isEnabled()

void putValue(String key, Object value)

Object getValue(String key)

void addPropertyChangeListener(PropertyChangeListener listener)

void removePropertyChangeListener(PropertyChangeListener listener)

public void actionPerformed(ActionEvent ie)

{

try

{

String ch="";

String s=(String)cb.getSelectedItem();

for(String st:choice)

{

if(((String)cb.getSelectedItem()).equals(st)

&& ((cb.getSelectedIndex())!=0))

{

ch = Integer.toString(counter);

data[i][j] = ch;

data[i][j+1] = ((String)cb.getSelectedItem());

for (k=1; k<10;k++)

{

if(((String)cb.getSelectedItem()).

equals(choice[k]))

n=k;

}

data[i][j+2]=price[n];

total=total+cost[n];

j=0;

counter++;

i++;

}

}

}

catch(Exception e)

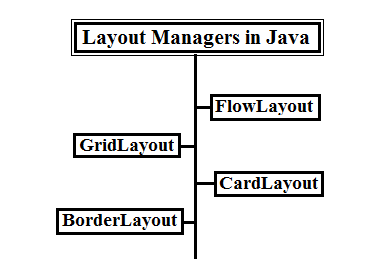
{

}

**LAYOUT MANAGERS IN JAVA**

Java class library provides us with pre-defined classes which are used for presenting controls according to some pre-defined presentation scheme these classes are technically referred to as LAYOUT MANAGERS.

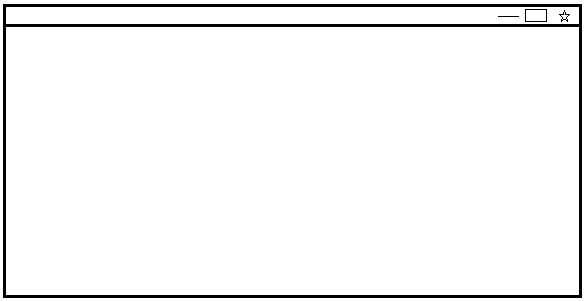
Java programming language/ AWT provides four layout managers class:



**FlowLayout:**

FlowLayout is a default layout manager which is applicable on panels. FlowLayout is a public class which add controls in a Flow from left to right in the order on their addition on the container object. Whenever we instantiate FlowLayout class, we are able to specify the alignment of controls.

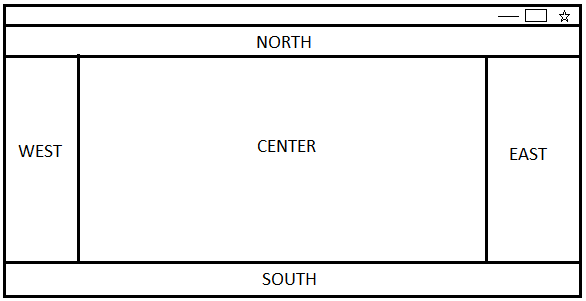
FlowLayout variable\_name= new FlowLayout(FlowLayout.RIGHT/ CENTER/ LEFT);



**BorderLayout:**

BorderLayout is a public class which is applicable on frame. BorderLayout divides the container in to 5 regions. Only one control can be added in a given region. At the time of adding control, we need to specify the region where the control is to be added.

BorderLayout variable\_name= new BorderLayout();

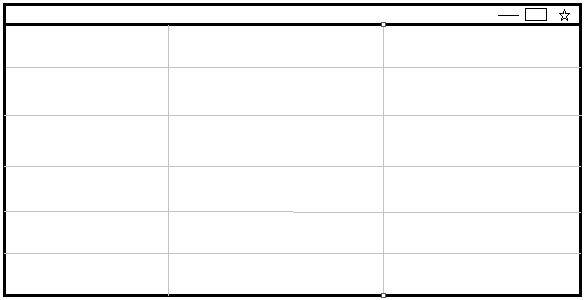


**GridLayout:**

GridLayout is a public class which displays controls in a tabular form application (in the forms of column and rows) we specify the total no of rows and columns at the time of instantiating GridLayout class.

GridLayout variable\_name= new GridLayout(no of rows, no of columns);

E.g.:- a grid layout with 6 columns and 3 rows



**CardLayout:**

CardLayout is a specialized layout manager which arranges and displays controls in the way cards present in a deck are arranged and displays. CardLayout is used when we want to display controls sequentially (one at a time) one after the other. CardLayout is a public class which provides following 4 methods for presenting controls added on a container object.

* next()
* first()
* last()
* previous()

CardLayout variable\_name= new CardLayout();

**EXCEPTION HANDLING**

In technical terms, a runtime error is referred to as EXCEPTION.

Every programming language provides a technique for handling runtime error/ exception that remedial programming is referred to as exception handling.

Exception handling is a special type of programming which ensures that the execution of program continues even if a runtime error has been generated during the execution of a program.

Exception handling prevents the program from reaching an abrupt end when a run time error is generated during the course of execution of the program.

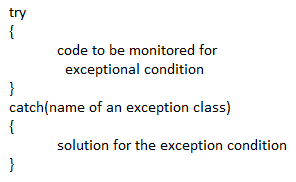
**Exception handling model in java programming language**

Java programming language provides a robust efficient and an extremely simple model for handling an exception/ run time.

Java programming language 5 keywords which can be used to handle exceptional condition originating from within an executing piece of code.

* Try
* Catch
* Finally
* Throw
* Throws

The most widely used mechanism for exception handling is to use try and catch block.



**Try** block is a block of code which contains the code from which an exceptional condition can be generated. In other words, try block contains the code which needs to be monitored for exceptional condition(s). Immediately after try block a block called catch defined.

**Catch** block contains the solution for the exceptional condition which might originate from within the try block. A catch block is executed only and only if an exception is generated from within the try block.

Catch block contains the name of an exception class and the argument. We can provide multiple catch block after a given try block. The catch block containing the solution to the exceptional condition generated from within the try block, will be executed.

**SYSTEM DESIGN**

The System Design phase will show to design planning and architectural planning of the project.

**DESIGN PLAN**

* Feasibility Study
* Requirement definition Analysis and Specification.

**ARCHITECTURE**

To develop the System for billing architecture is maintained and the model will work with standalone application. The coding and implementation resides on the same machine. Employee working on the application perform different operations and manipulate the request and generate the result as per customers’ requirements.

**Typical role of employee side:-**

* The employee should gather all needed information before making a final bill.
* The employee is responsible for all display of data to the customer.
* The employee should handle all data manipulation operations.
* The employee provides all formatting of data and presentation of information in reports.

**DATA MODEL**

As per requirement the information that is to be stored in coding is:-

* Information about product(s) (like name, price, storage, version etc.)
* Login details (like user name, password etc.)
* It improves a multi-user updating through a GUI front end.

**ARCHITETCTURAL DESIGN**

**Operational and Functional Procedural Description**

The complete system is developed using modular development technique. The task is divided into four major modules, namely:

* **Login**
* **Add Product(s)**
* **Delete Product(s)**
* **Modify Product(s)**
* Login Module:This module takes user name and password. This module validates username and password.

Input parameter :user name, password

Output parameters:Permission Accepted / Denied

* Add Product(s) Module:This process is used to add the products and updates the coding accordingly.

Input parameter :name, price

* Erase Product(s) Module:This process is used to erase the product(s) and updates the coding accordingly.

Input parameter :username

Output parameters :product deleted from coding

* Modify product(s) Module:This process is used to change any record already present in the coding and updates the coding accordingly.

Input parameter :name, price of the product

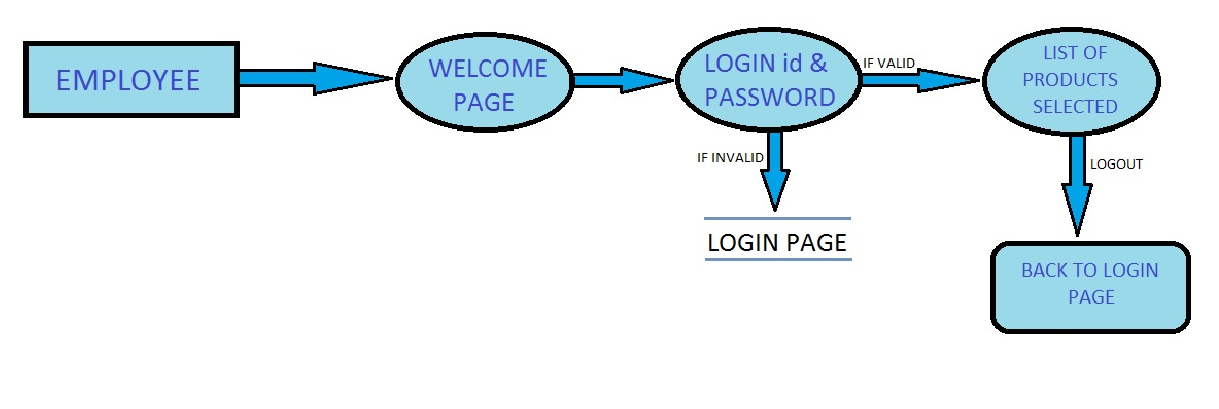
Output parameters :product added to coding

**DATA FLOW DIAGRAMS**

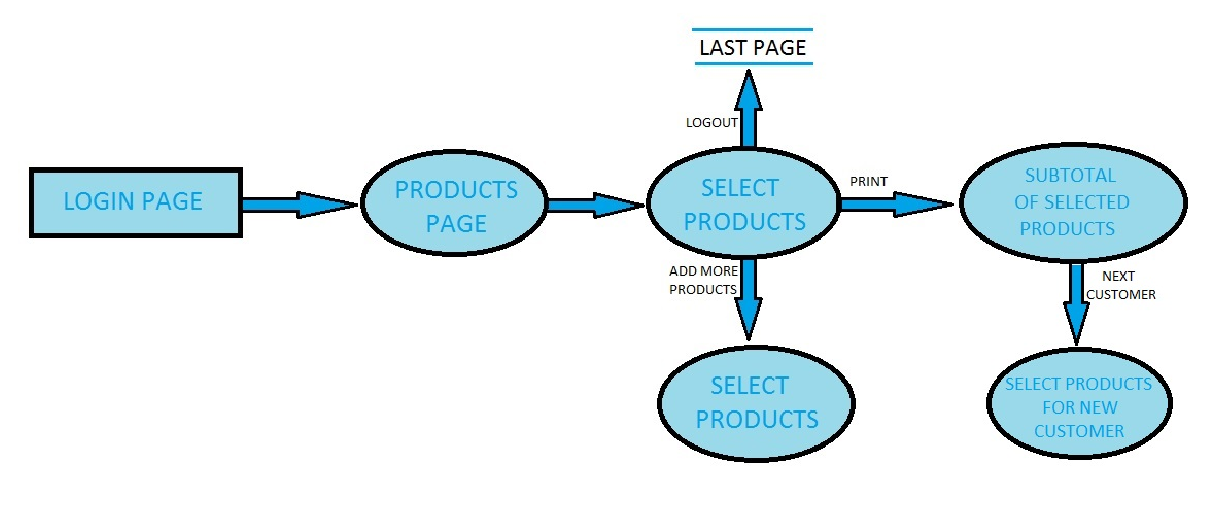
A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel unlike a flowchart which also shows this information

**DATA FLOW DIAGRAM FOR EMPLOYEES LOGIN**

****After entering to the home page of the application, employee can choose the LOGIN option where they are asked to enter username & password, and if he/she is a valid user then a list of products page will be displayed. If the employee enters an invalid password then a pop-up message will be displayed containing a message to enter a valid user id or password. After selection of the product(s) as per the requirements of the customer(s), a bill is generated. On logout the final page of the application will appear thanking the customer for visit.

**DATA FLOW DIAGRAM FOR BUYING PRODUCT(S)**

****

It is a 2nd level Data Flow Diagram where after entering the LOGIN page the employee can select the product(s) as selected by the customer. A list of selected products will be displayed with in the form of table containing SERIAL NO, PRODUCT NAME and PRICE of the product(s) selected. Total amount to be paid will be mentioned at the right bottom corner of the screen. After the payment is done the employee can opt for an option named NEXT CUSTOMER after which the whole process of selection and billing of products can be executed.

**CODING OF THE APPLICATION**

/\*\*

\* The techno program implements an application that

\* can be used as billing software.

\*

\* @authors Poonam Verma

\* Ankit Rohilla

\* @version 1.0

\*/

import java.awt.\*;

import java.io.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.JTable;

class techno implements MouseListener,ActionListener

{

int i=0;

int j=0,k,n=0,total=0;

int counter=1;

JFrame f;

JPanel p1,p2,p3,p4,p5,p6;

JPanel scr2,east,west,p21,p22,p23;

JPanel bp1,bp2,bp3,east1,west1;

JButton pro,log,clr,lout,adp,pri,bac,paid;

Label l1,l2,l3,l4,empty,empty1,l5,l6;

TextField tf1,tf2,tf3;

Font font,font2,font3,font4;

GridLayout gl,gl2;

BorderLayout bl,b2;

JScrollPane sp;

JTable jt;

String login,psw,s1,s2;

String[] choice={"Select..","WD\_HDD\_1tb",

"SANDISK PD 32gb",

"SEGABYTE HDD 4TB",

"HP MOUSE",

"ASUS gtx",

"LOGITEC KEYBOARD",

"INTEX HEADPHONES",

"DVD DRIVE",

"KEYBOARD+MOUSE COMBO",

"JOYSTICK"};

String[] price={"0","5999.00","380.00","8655.00","499.00",

"10999.00","799.00","1400.00","1199.00","900.00","800.00"};

int[] cost={0,5999,380,8655,499,10999,799,1400,1199,900,800};

JComboBox cb;

String data[][]=new String[20][3];

JMenuBar m;

JMenu m1,m2;

public techno()

{

font=new Font("SansSerif",Font.BOLD,20);

font2=new Font("SansSerif",Font.BOLD,32);

font3=new Font("Times New Roman",Font.BOLD,15);

font4=new Font("Times New Roman",Font.BOLD,14);

**//First screen of the application**

gl=new GridLayout(2,1);

f=new JFrame();

f.setVisible(true);

l1=new Label(" Welcome To TechnoShop");

l1.setFont(font);

pro=new JButton("PROCEED");

pro.setPreferredSize(new Dimension(100,40));

p1=new JPanel();

p5=new JPanel();

p6=new JPanel();

p1.setLayout(gl);

JLabel logo=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\ts.jpg"));

logo.setLayout(new FlowLayout());

p6.add(logo);

p1.add(p6);

p6.add(l1);

p1.add(p5);

p5.add(pro);

f.add(p1);

p5.setBackground(new Color(77,180,180));

m=new JMenuBar();

m1=new JMenu("About Us");

m2=new JMenu("Contact Us");

m.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);

m.add(m2);

m.add(m1);

f.setJMenuBar(m);

**//Second screen of the application**

p2=new JPanel();

p21=new JPanel();

p22=new JPanel();

p23=new JPanel();

scr2=new JPanel();

east=new JPanel();

west=new JPanel();

f.add(p1);

p1.setVisible(true);

gl2=new GridLayout(3,2);

b2=new BorderLayout();

l2=new Label("LOGIN\_ID");

l2.setFont(font3);

l3=new Label("PASSWORD");

l3.setFont(font3);

log=new JButton("LOGIN");

clr=new JButton("CLEAR");

tf1=new TextField(15);

tf1.setFont(font3);

tf2=new TextField(15);

tf2.setFont(font3);

tf2.setEchoChar('\*');

scr2.setLayout(b2);

scr2.add(p2,"Center");

p2.setLayout(gl2);

p2.add(p21);

p21.setBackground(new Color(77,169,180));

p2.add(p22);

p22.setBackground(new Color(77,172,180));

p2.add(p23);

p23.setBackground(new Color(77,177,180));

p21.add(l2);

p21.add(tf1);

p22.add(l3);

p22.add(tf2);

p23.add(log);

p23.add(clr);

scr2.add(east, “East");

JLabel logo1=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\pic31.jpg"));

logo1.setLayout(new FlowLayout());

east.add(logo1);

scr2.add(west, “West");

JLabel logo2=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\pic41.jpg"));

logo2.setLayout(new FlowLayout());

west.add(logo2);

**//Third screen of the application**

p3=new JPanel();

bp1=new JPanel();

bp2=new JPanel();

bp3=new JPanel();

bl=new BorderLayout();

l4=new Label("PRODUCTS");

l4.setFont(font4);

empty=new Label(" ");

empty1=new Label(" ");

l5=new Label("SUB\_TOTAL");

l5.setFont(font4);

lout=new JButton("LOGOUT");

lout.setFont(font4);

adp=new JButton("ADD MORE PRODUCTS");

adp.setFont(font4);

pri=new JButton("PRINT");

pri.setFont(font4);

paid=new JButton("NEXT CUSTOMER");

paid.setFont(font4);

tf3=new TextField(15);

cb=new JComboBox(choice);

p3.setLayout(bl);

p3.add(bp1,"North");

bp1.add(paid);

bp1.add(empty1);

bp1.add(l4);

bp1.add(cb);

bp1.add(lout);

p3.add(bp2,"South");

bp2.add(adp);

bp2.add(pri);

p3.add(bp3,"Center");

bp3.setBackground(new Color(77,120,250));

bp2.setBackground(Color.lightGray);

bp2.add(empty);

bp2.add(l5);

bp2.add(tf3);

**//Last screen of the application**

p4=new JPanel();

east1=new JPanel();

west1=new JPanel();

p4.setLayout(new BorderLayout());

JLabel background=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\ts.jpg"));

p4.add(background,"Center");

background.setLayout(new FlowLayout());

l6=new Label(" Have A Nice Day!");

p4.add(l6,"North");

l6.setBackground(new Color(77,177,180));

l6.setFont(font2);

bac=new JButton("BACK TO LOGIN PAGE");

bac.setFont(font3);

p4.add(bac,"South");

JLabel logo3=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\second.jpg"));

logo3.setLayout(new FlowLayout());

west1.add(logo3);

p4.add(west1,"West");

JLabel logo4=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\first.jpg"));

logo4.setLayout(new FlowLayout());

east1.add(logo4);

p4.add(east1,"East");

f.setLocation(250,160);

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

f.setSize(880,390);

f.setResizable(false);

f.setTitle("TechnoShop BILLING SYSTEM");

f.setVisible(true);

**//Registering MouseListener & ActionListener**

pro.addMouseListener(this);

log.addMouseListener(this);

clr.addMouseListener(this);

adp.addMouseListener(this);

lout.addMouseListener(this);

bac.addMouseListener(this);

pri.addMouseListener(this);

cb.addActionListener(this);

m1.addMouseListener(this);

m2.addMouseListener(this);

paid.addMouseListener(this);

}

**//Event Handling**

public void mouseClicked(MouseEvent me)

{

if(me.getSource()==pro)

{

p1.setVisible(false);

scr2.setVisible(true);

f.add(scr2);

f.setVisible(true);

}

else if(me.getSource()==log)

{

login="techno2017";

psw="ts0123";

s1=tf1.getText();

s2=tf2.getText();

if(s1.equals(login)&& s2.equals(psw))

{

scr2.setVisible(false);

p3.setVisible(true);

f.add(p3);

f.setVisible(true);

JOptionPane.showMessageDialog(f,"Please select

the product(s) from the Dropdown List. \nThen press PRINT");

tf1.setText(null);

tf2.setText(null);

}

else

{

JOptionPane.showMessageDialog(f,"Access Denied. \n You have entered an invalid username/password");

}

}

else if(me.getSource()==clr)

{

try

{

p2.setVisible(true);

scr2.add(p2);

f.add(scr2);

f.setVisible(true);

tf1.setText(null);

tf2.setText(null);

}

catch(Exception e)

{

}

}

else if(me.getSource()==lout)

{

p3.setVisible(false);

if(n!=0)

{

sp.setVisible(false);

sp.getViewport().remove(jt);

}

pri.setText("PRINT");

p4.setVisible(true);

f.add(p4);

f.setVisible(true);

}

else if(me.getSource()==bac)

{

p4.setVisible(false);

data=new String[10][3];

cb.removeAllItems();

i=0;

counter=1;

total=0;

tf3.setText("");

scr2.setVisible(true);

for(int i=0;i<10;i++)

cb.addItem(choice[i]);

f.add(scr2);

f.setVisible(true);

}

else if(me.getSource()==pri)

{

if(n!=0)

{

String column[]={"S.NO",

"PRODUCT",

"PRICE"};

jt=new JTable(data, column);

jt.setBounds(30,40,200,300);

sp=new JScrollPane(jt);

bp3.add(sp);

bp3.setVisible(true);

f.setVisible(true);

String tot=Integer.toString(total);

tf3.setText(tot+".00");

}

else

JOptionPane.showMessageDialog(f,

"Please select at least one product.");

}

else if(me.getSource()==adp)

{

sp.getViewport().remove(jt);

pri.setText("RE-PRINT");

JOptionPane.showMessageDialog(f,

"Select other items you want.\nThen press RE-PRINT");

p3.setVisible(true);

cb.removeAllItems();

for(int i=0;i<10;i++)

cb.addItem(choice[i]);

f.add(p3);

f.setVisible(true);

}

else if(me.getSource()==m1)

{

JOptionPane j=new JOptionPane();

j.showOptionDialog(null,"Hello! This is TechnoShop v1.0 \n

A Technological Shop to offer customers a better service and build loyalty.\n\n

This program is free software; you can redistribute it and/or modify it under the\n

terms of the GNU General Public License as published by the Free Software Foundation;\n

either version 2 of the License, or (at your option) any later version.\n\n

Authors:\nPOONAM VERMA\n

ANKIT ROHILLA",

"About TechnoShop", j.DEFAULT\_OPTION,j.INFORMATION\_MESSAGE, null, new Object[]{}, null);

}

else if(me.getSource()==m2)

{

JOptionPane y=new JOptionPane();

y.showOptionDialog(null,"technoShop2k17@gmail.com \n\n(+91)9876543210 \n(+91)8978675645","Contact Us",

y.DEFAULT\_OPTION,y.INFORMATION\_MESSAGE,

null, new Object[]{}, null);

}

else if(me.getSource()==paid)

{

p3.setVisible(false);

i=0;

j=0;

if(n!=0)

{

sp.setVisible(false);

sp.getViewport().remove(jt);

total=0;

}

cb.removeAllItems();

tf3.setText(null);

for(int i=0;i<10;i++)

cb.addItem(choice[i]);

pri.setText("PRINT");

for(int i=0;i<10;i++)

{

for(int j=0;j<3;j++)

data[i][j]=null;

}

counter=1;

p3.setVisible(true);

f.add(p3);

f.setVisible(true);

}

}

public void mouseReleased(MouseEvent me)

{

}

public void mouseEntered(MouseEvent me)

{

}

public void mouseExited(MouseEvent me)

{

}

public void mousePressed(MouseEvent me)

{

}

public void actionPerformed(ActionEvent ie)

{

try

{

String ch="";

String s=(String)cb.getSelectedItem();

for(String st:choice)

{

if(((String)cb.getSelectedItem()).equals(st)

&& ((cb.getSelectedIndex())!=0))

{

ch=Integer.toString(counter);

data[i][j]=ch;

data[i][j+1]=((String)cb.getSelectedItem());

for(k=1;k<10;k++)

{

if(((String)cb.getSelectedItem()).equals(choice[k]))

n=k;

}

data[i][j+2]=price[n];

total=total+cost[n];

j=0;

counter++;

i++;

}

}

}

catch(Exception e)

{

}

}

**//main() Class defined**

public static void main(String args[])

{

techno t=new techno();

}

}

**SYSTEM TESTING**

The aim of the system testing process was to determine all defects in our project .The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing

2. Integration testing

**UNIT TESTING**

Unit testing is undertaken when a module has been created and successfully reviewed.

In order to test a single module we need to provide a complete environment i.e. besides the module we would require

* The procedures belonging to other modules that the module under test calls.
* Non local data structures that module accesses.
* A procedure to call the functions of the module under test with appropriate parameters.

**TEST FOR EMPLOYEE MODULE**

* Testing employee login form-

This form is used for log in of staff of the system. In this we enter the username and password if both are correct administration page will open otherwise if any of data is wrong it will get redirected back to the login page and again ask for username and password.

* Product addition-

In this section the admin can verify product(s) details from product(s) label and then only add product details to main coding. The user has to save the coding after adding more products and have to compile again.

* Price Addition-

Employee can enter price of product(s) and can he can view the product(s) in the output.

**INTEGRATION TESTING**

In this type of testing we test various integration of the project module by providing the input .The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **:** | This test will check the application **"TechnoShop BILLING SYSTEM"** |
|  |  |  |
| **Test Procedure** | **:** | To Log In: Enter User Name, Password & Let your default login name & password be techno2017, ts0123. |
|  |  |  |
| **Expected Result** | **:** | If login is validated you are allowed to go to next page. |
|  |  |  |
| **Actual Result** | **:** | Next page show the main menu page where different task can be performed like selection of products, addition of more products and calculation of Sub-Total of the products. |
|  |  |  |
| **Comment** | **:** | Login information is checked. |
|  |  |  |
| **Conditional Test** | **:** | If the user enters wrong password or user name. |
|  |  |  |
| **Expected Result** | **:** | The system will show an error message.  "Access Denied. Please enter valid user name & password" |
|  |  |  |
| **Actual Result** | **:** | The error message shown to the user prompt user to login again |
|  |  |  |
| **Accuracy** | **:** | The working is accurate. It is running properly in operational software. |

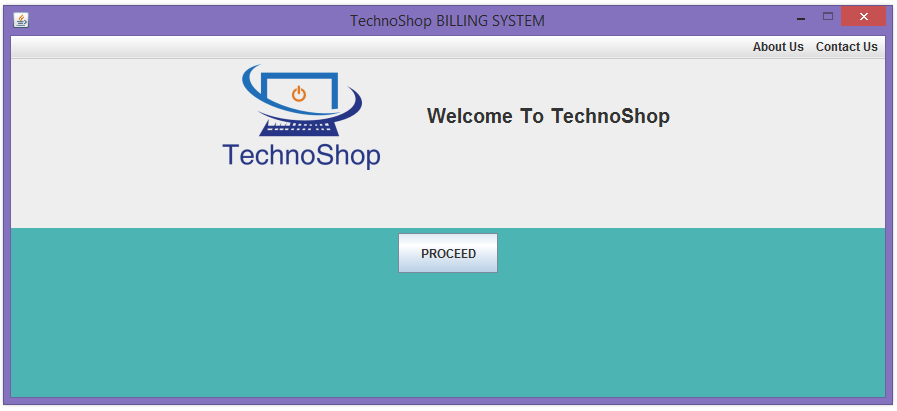
**SYSTEM IMPLEMENTATION**

**SCREENSHOTS OF THE APPLICATION**

The **FIRST PAGE** of the application contains controls and layout manger i.e. GridLayout of AWT (abstract window toolkit).

It contains the following controls:

* Frame
* Panel
* Label
* Button
* Menu
* MenuBar

****

**Coding for first page:**

gl=new GridLayout(2,1);

f=new JFrame();

f.setVisible(true);

l1=new Label(" Welcome To TechnoShop");

l1.setFont(font);

pro=new JButton("PROCEED");

pro.setPreferredSize(new Dimension(100,40));

p1=new JPanel();

p5=new JPanel();

p6=new JPanel();

p1.setLayout(gl);

JLabel logo=new JLabel(new ImageIcon("C:\\Users\\hp\\Desktop\\PROJECT\\ts.jpg"));

logo.setLayout(new FlowLayout());

p6.add(logo);

p1.add(p6);

p6.add(l1);

p1.add(p5);

p5.add(pro);

f.add(p1);

p5.setBackground(new Color(77,180,180));

m=new JMenuBar();

m1=new JMenu("About Us");

m2=new JMenu("Contact Us");

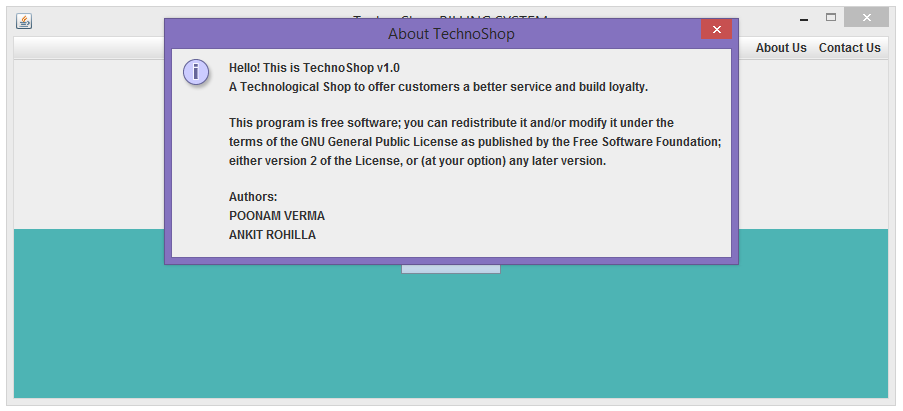
m.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);

m.add(m2);

m.add(m1);

f.setJMenuBar(m);

On clicking **ABOUT US** menu in the menu bar the following dialog box will appear containing the information about TechnoShop.

****

**Code for the dialog box:**

JOptionPane j=new JOptionPane();

j.showOptionDialog (null,"Hello! This is TechnoShop v1.0 \n

A Technological Shop to offer customers a better service and build loyalty.\n\n

This program is free software; you can redistribute it and/or modify it under the\n

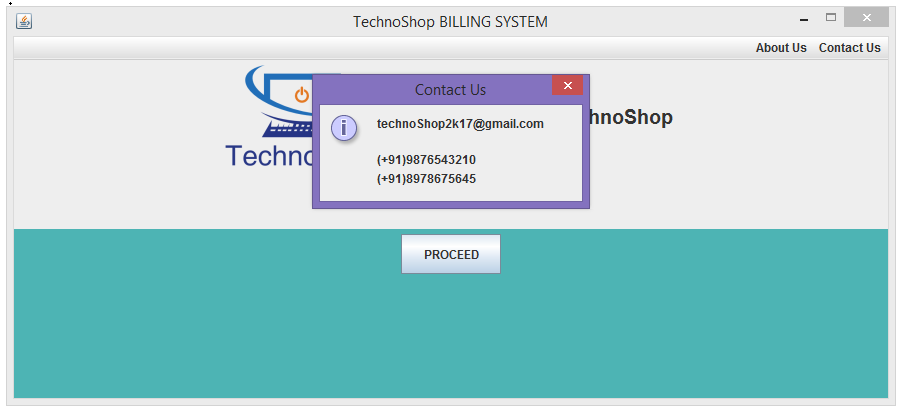
terms of the GNU General Public License as published by the Free Software Foundation;\neither version 2 of the License, or (at your option) any later version.\n\n

Authors:\n

POONAM VERMA\n

ANKIT ROHILLA","About TechnoShop", j.DEFAULT\_OPTION,j.INFORMATION\_MESSAGE, null, new Object[]{}, null);

On clicking the **CONTACT US** menu in the menu bar the following dialog box will appear containing the information how a customer can contact the shop authority.



**Code for the dialog box:**

JOptionPane y=new JOptionPane();

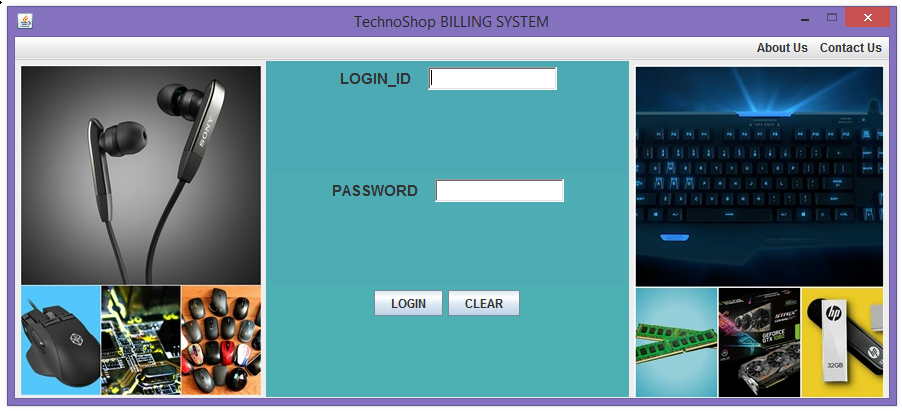
y.showOptionDialog(null,"technoShop2k17@gmail.com\n\n

(+91)9876543210 \n

(+91)8978675645","Contact Us",

y.DEFAULT\_OPTION,y.INFORMATION\_MESSAGE, null, new Object[]{}, null);

The **SECOND SCREEN** appears when we click the PROCEED button. On clicking the button first panel will disappear and a new panel will appear. On this panel we have added AWT controls and layout managers.



**Coding for the second panel:**

p2=new JPanel();

p21=new JPanel();

p22=new JPanel();

p23=new JPanel();

scr2=new JPanel();

east=new JPanel();

west=new JPanel();

f.add(p1);

p1.setVisible(true);

gl2=new GridLayout(3,2);

b2=new BorderLayout();

l2=new Label("LOGIN\_ID");

l2.setFont(font3);

l3=new Label("PASSWORD");

l3.setFont(font3);

log=new JButton("LOGIN");

clr=new JButton("CLEAR");

tf1=new TextField(15);

tf2=new TextField(15);

tf2.setEchoChar('\*');

scr2.setLayout(b2);

scr2.add(p2,"Center");

p2.setLayout(gl2);

p2.add(p21);

p21.setBackground(new Color(77,169,180));

p2.add(p22);

p22.setBackground(new Color(77,172,180));

p2.add(p23);

p23.setBackground(new Color(77,177,180));

p21.add(l2);

p21.add(tf1);

p22.add(l3);

p22.add(tf2);

p23.add(log);

p23.add(clr);

scr2.add(east,"East");

JLabel logo1=new JLabel(newImageIcon

("C:\\Users\\hp\\Desktop\\PROJECT\\pic31.jpg"));

logo1.setLayout(new FlowLayout());

east.add(logo1);

scr2.add(west,” West");

JLabellogo2=newJLabel(newImageIcon

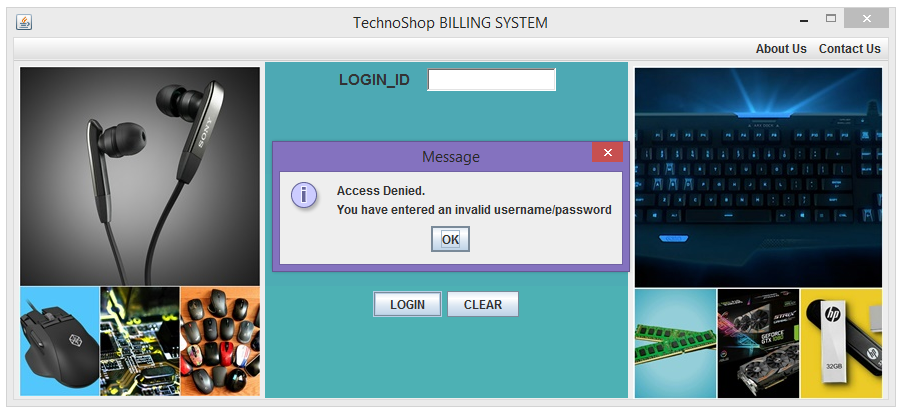
("C:\\Users\\hp\\Desktop\\PROJECT\\pic41.jpg"));

logo2.setLayout(new FlowLayout());

west.add(logo2);

The **THIRD SCREEN** appears when we click the LOGIN button. When we click it the second panel will disappear and a new panel will appear containing Login Id and Password fields. We have added AWT controls and layout managers.

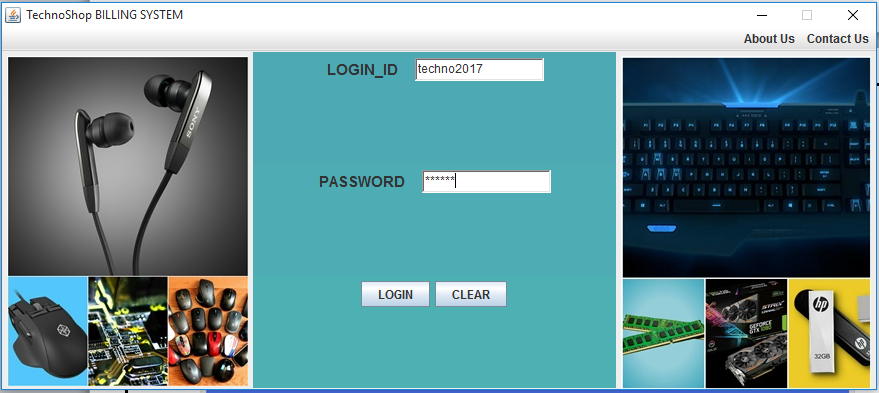
* If the User enters anything inappropriate in the Login\_Id or password column, he can press the CLEAR button
* If User Id or Password is **invalid** it will display a dialog box containing a message “Access Denied. You have entered an invalid username/password ”.

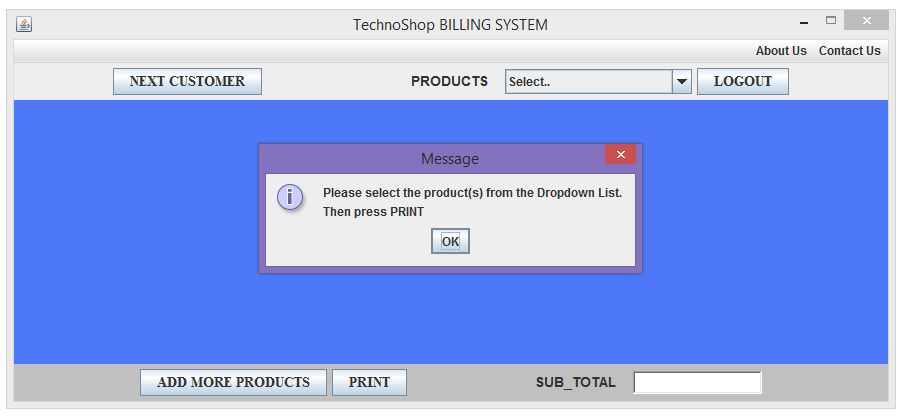


JOptionPane.showMessageDialog (f,"Access Denied. \n

You have entered an invalid username/password");

* If the user enters a **valid** User Id and Password then a new panel will appear with a dialog box containing the message: “Please select the product(s) from the Dropdown list and then press PRINT”.





**Code for the third screen:**

[FOR DIALOG BOX

JOptionPane.showMessageDialog(f,"Please select the product(s) from the

Dropdown List. \nThen press PRINT");

]

p3=new JPanel();

bp1=new JPanel();

bp2=new JPanel();

bp3=new JPanel();

bl=new BorderLayout();

l4=new Label("PRODUCTS");

l4.setFont(font4);

empty=new Label(" ");

empty1=new Label(" ");

l5=new Label("SUB\_TOTAL");

l5.setFont(font4);

lout=new JButton("LOGOUT");

lout.setFont(font4);

adp=new JButton("ADD MORE PRODUCTS");

adp.setFont(font4);

pri=new JButton("PRINT");

pri.setFont(font4);

paid=new JButton("NEXT CUSTOMER");

paid.setFont(font4);

tf3=new TextField(15);

cb=new JComboBox(choice);

p3.setLayout(bl);

p3.add(bp1,"North");

bp1.add(paid);

bp1.add(empty1);

bp1.add(l4);

bp1.add(cb);

bp1.add(lout);

p3.add(bp2,"South");

bp2.add(adp);

bp2.add(pri);

p3.add(bp3,"Center");

bp3.setBackground(new Color(77,120,250));

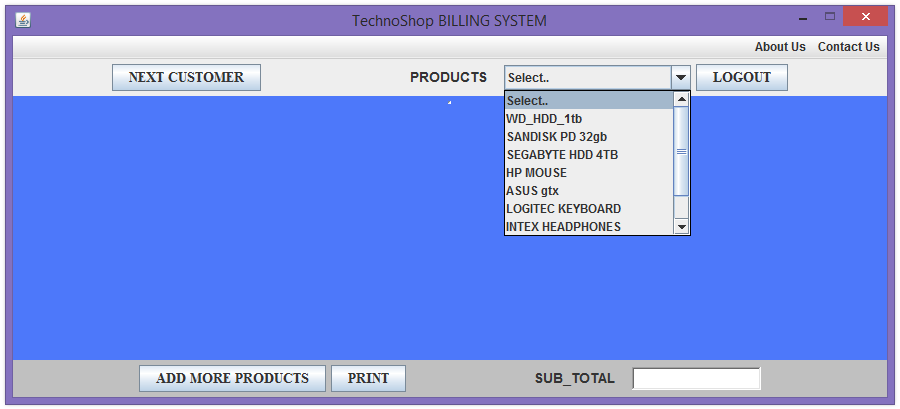
bp2.setBackground(Color.lightGray);

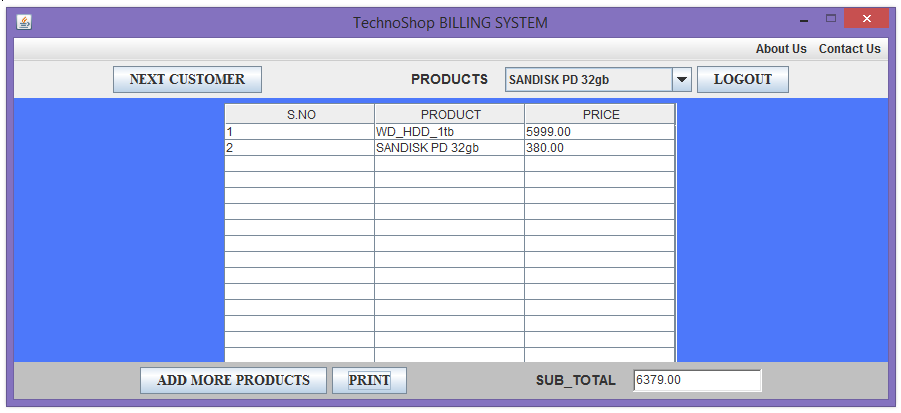
bp2.add(empty);

bp2.add(l5);

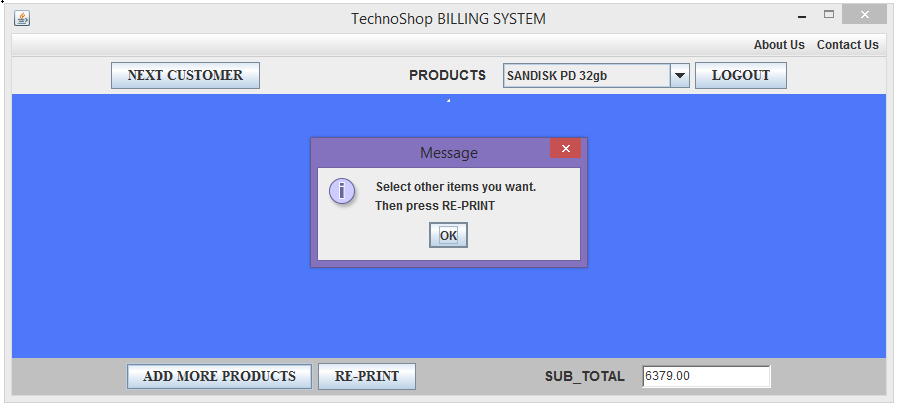
bp2.add(tf3);

After the employee logs into the system the customer can request him for buying the products. The employee selects the items which the customer wants to buy and a list of all the selected items along with its price is displayed in the form of a table. Subtotal of the products is calculated and displayed at the right bottom of the screen.

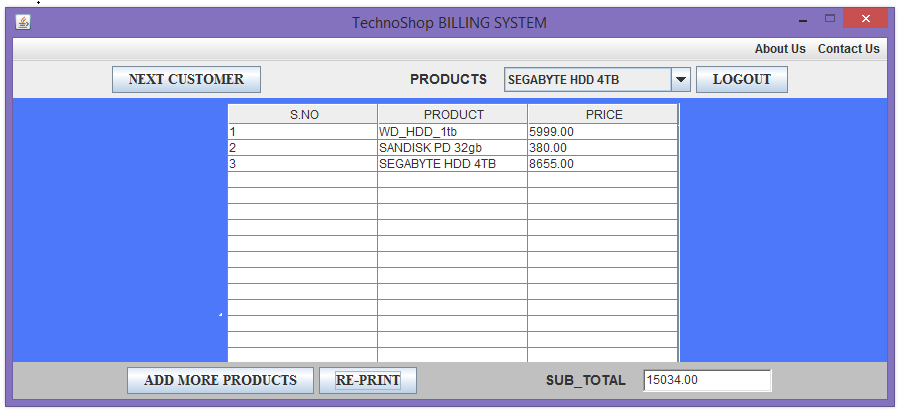




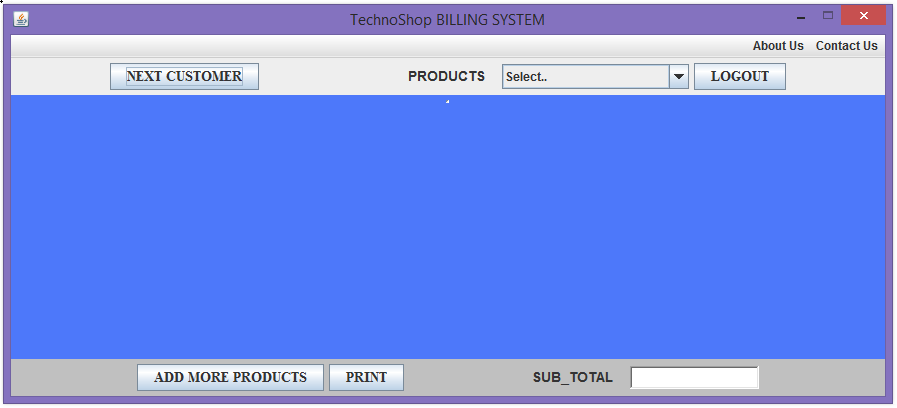
If the customer wants to buy more product he/ she can ask the employee. On clicking **ADD MORE PRODUCTS** button a dialog box will appear containing “Select other items you want. Then press RE-PRINT”.

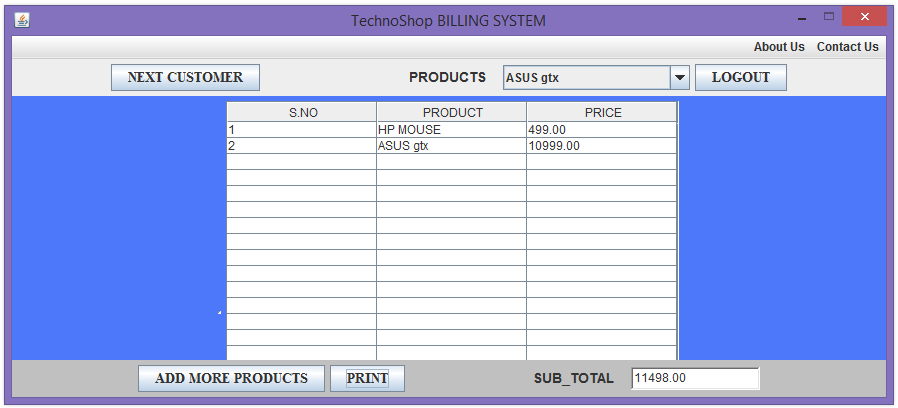


After selecting all the products the customer wants the Sub Total of the products will be displayed in a Text Field on the right bottom side of the screen.



When the transaction of the first customer is completed the employee can click on “**NEXT CUSTOMER**” button and make the transactions for the second customer.





The **FOURTH SCREEN** appears when we click on LOGOUT button. When we click the button the third panel will disappear and the new panel will appear containing a button. We added AWT controls and layout managers.



**Code for the fourth screen i.e. the Last Screen**

p4=new JPanel();

east1=new JPanel();

west1=new JPanel();

p4.setLayout(new BorderLayout());

JLabelbackground=newJLabel(newImageIcon(

"C:\\Users\\hp\\Desktop\\PROJECT\\ts.jpg"));

p4.add(background,"Center");

background.setLayout(new FlowLayout());

l6=new Label(" Have A Nice Day!");

p4.add(l6,"North");

l6.setBackground(new Color(77,177,180));

l6.setFont(font2);

bac=new JButton("BACK TO LOGIN PAGE");

bac.setFont(font3);

p4.add(bac,"South");

JLabellogo3=newJLabel(newImageIcon(

"C:\\Users\\hp\\Desktop\\PROJECT\\second.jpg"));

logo3.setLayout(new FlowLayout());

west1.add(logo3);

p4.add(west1,"West");

JLabellogo4=newJLabel(newImageIcon(

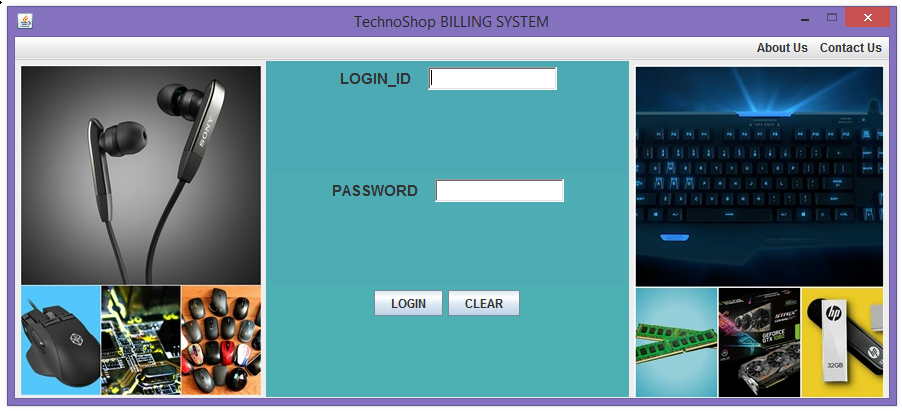
"C:\\Users\\hp\\Desktop\\PROJECT\\first.jpg"));

logo4.setLayout(new FlowLayout());

east1.add(logo4);

p4.add(east1,"East");

The **FOURTH SCREEN** has a button named BACK TO LOGIN PAGE. On pressing this button the employee is again directed to the LOGIN PAGE. He can again Login with a valid Id and Password if required.



On clicking the CLOSE button present in the title bar, the application closes. This is achieved by a special function in java:

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

**CONCLUSION**

This system provides a computerized version of billing management system which will benefit the employee as well as the staff of the shop. It makes entire process computerized; staff can generate reports and do bill transactions. It also has a facility for customers where they can see status of bill of products which they are buying as well request staff members to add more products or can give some suggestions.

Development of this system takes lots of efforts. We think the system gave a lot of satisfaction. Though every task is never said to be perfect in this development field even more improvement may be possible in this system. We learnt so many things and gained a lot of knowledge about development field. We hope this prove fruitful.

**FUTURE SCOPE**

There is a future scope of this facility that many more features such as online order system can be added. The customers can buy the products from home and the products will be delivered in their address. A feature of data base can be added so that all the bills can be saved in the data base along with the name of the customer and their address. And a feature can also be made that every customer have their own id and password thus making it more interactive more user friendly and project which fulfils each users need in the best way possible.

* This project will help the store employee in fast billing.
* This project will enable to see report regarding products and category.
* It is easy to maintain in future prospect.

**BIBLIOGRAPHY**

* JAVA 2 CORE LANGUAGE LITTLE BLACK BOOK BY ALAIN TROTTIER.
* CORE JAVA- AN INTEGRATED APPROACH BY R.NAGESWARA RAO.
* CORE JAVA 2: VOLUME I–FUNDAMENTALS BY CAY S. HORSTMANN