**Classes:**

Class ShopApplet

**/\***

**<applet code="ui\ShopApplet.class" height="607" width="1080">**

**</applet>**

**\*/**

package ui;

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.\*;

import java.beans.\*;

import java.util.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import product.\*;

import product.ProductList.Category;

public class ShopApplet extends Applet implements ActionListener {

private final int HOME = 1;

private final int SEARCH = 2;

private final int CART = 3;

private final int DELIVERY\_OPTIONS = 4;

private final int PAYMENT = 5;

private final String CVV = "453";

private final String FILTER\_MSG = "Nothing Found... Try changing filter selection";

private final String CART\_MSG = "Cart Empty :( Start Shopping...";

private final String HOME\_MSG = "Done for today :)";

private final String SEARCH\_LONG\_MSG = "Nothing Found... Try Typing a longer word...";

private int page = HOME;

private double total = 0d;

private int extra = 0;

private String key = "";

private ProductList db = ProductList.getDatabase();

private ProductList shopItems = new ProductList(db);

private ProductList cartItems = new ProductList();

private ProductList purchaseItems = new ProductList();

private ProductList featuredItems = new ProductList();

private Vector<Integer> selectedCat = new Vector<>();

private ImageIcon backIcon;

private ImageIcon cartIcon;

private ImageIcon addIcon;

private ImageIcon removeIcon;

private ImageIcon checkOutIcon;

private ImageIcon payIcon;

private Container shop = new Container();

private Container navBar = new Container();

private JButton back;

private JTextField searchBar = new JTextField("Search...");

private JLabel title = new JLabel();

private JButton action;

private JButton next;

private Container content = new Container();

private Container sideMenu = new Container();

private JLabel sideTitle = new JLabel("Filters");

private JComboBox<String> sortOption = new JComboBox<>();

private JCheckBox[] catOptions = new JCheckBox[7];

private JButton reset = new JButton("Reset");

private Container mainContainer = new Container();

private Container featured = new Container();

private JLabel featuredLabel = new JLabel("Featured Items");

private JList<Product> featuredItemsList = new JList();

private JScrollPane featuredPanel = new JScrollPane(featuredItemsList);

private JLabel appLabel = new JLabel("Shop Items");

private JList<Product> shopItemList = new JList<>();

private JScrollPane panel = new JScrollPane(shopItemList);

private JLabel emptyMSG = new JLabel();

private Container deliveryContainer = new Container();

private JRadioButton standardButton = new JRadioButton("Standard Delivery-Expect package to be delivered in 7 working days");

private JRadioButton expressButton = new JRadioButton("Express Delivery-Expect package to be delivered tomorrow: +Rs 250");

private Container paymentConatiner = new Container();

private JLabel ccInfo = new JLabel("Credit Card Information");

private JLabel sccNumber = new JLabel("Saved CC Number:");

private JLabel ccNumber = new JLabel("XXXX-XXXX-XXXX-6540");

private JLabel enterCC = new JLabel("Enter CVV:");

private JPasswordField cvvField = new JPasswordField(3);

private Container bottomBar = new Container();

private JButton bNext;

private JLabel summary = new JLabel();

private MouseListener l = new MouseAdapter() {

long p = 0;

public void mouseClicked(MouseEvent me) {

if (me.getButton() == MouseEvent.BUTTON1)

if (System.currentTimeMillis() - p < 300)

actionPerformed(new ActionEvent(action, 0, null));

else p = System.currentTimeMillis();

}

};

private Dimension screenSize;

public void init() {

screenSize = Toolkit.getDefaultToolkit().getScreenSize();

System.out.println(screenSize.getHeight());

setSize((int) (screenSize.getWidth() \* .95), (int) (screenSize.getHeight() \* .8));

backIcon = new ImageIcon(getImage(getCodeBase(), "back.png"));

cartIcon = new ImageIcon(getImage(getCodeBase(), "cart.png"));

addIcon = new ImageIcon(getImage(getCodeBase(), "add.png"));

removeIcon = new ImageIcon(getImage(getCodeBase(), "remove.png"));

checkOutIcon = new ImageIcon(getImage(getCodeBase(), "check\_out.png"));

payIcon = new ImageIcon(getImage(getCodeBase(), "pay.png"));

back = new JButton(backIcon);

next = new JButton("Cart (0)", cartIcon);

action = new JButton(addIcon);

bNext = new JButton(payIcon);

back.setToolTipText("Back");

sideTitle.setFont(new Font(null, Font.BOLD, 13));

sideTitle.setForeground(Color.WHITE);

featuredLabel.setFont(new Font(null, Font.BOLD, 18));

featuredLabel.setForeground(Color.WHITE);

appLabel.setFont(new Font(null, Font.BOLD, 18));

appLabel.setForeground(Color.WHITE);

cvvField.addKeyListener(new KeyListener() {

public void keyTyped(KeyEvent ke) {

if ((int) ke.getKeyChar() == KeyEvent.VK\_ENTER)

actionPerformed(new ActionEvent(bNext, 0, null**));**

if (!Character.isDigit(ke.getKeyChar()) || cvvField.getText().length() == 3)

ke.consume();

}

public void keyPressed(KeyEvent ke) {}

public void keyReleased(KeyEvent ke) {}

});

for (JLabel l : new JLabel[]{title, summary}) {

l.setFont(new Font(null, Font.BOLD, 24));

l.setForeground(Color.WHITE);

}

for (JButton b : new JButton[]{next, back, reset, action, bNext}) {

b.setBackground(Color.decode("#1976D2"));

b.setForeground(Color.decode("#ffffff"));

b.addActionListener(this);

}

searchBar.addKeyListener(new KeyListener() {

public void keyTyped(KeyEvent ke) { }

public void keyPressed(KeyEvent ke) {}

public void keyReleased(KeyEvent ke) {

if (page == CART) initHome();

if (page != SEARCH) {

featuredLabel.setVisible(false);

featuredPanel.setVisible(false);

}

page = SEARCH;

key = searchBar.getText();

sideMenu.setVisible(true);

if (ke.getKeyCode() == KeyEvent.VK\_ESCAPE) initHome();

updateList();

if (shopItems.isEmpty())

if (key.length() < 3) emptyMSG.setText(SEARCH\_LONG\_MSG);

else emptyMSG.setText("Nothing Found for \"" + key + "\"");

}

});

searchBar.addFocusListener(new FocusListener() {

public void focusGained(FocusEvent fe) {

searchBar.selectAll();

}

public void focusLost(FocusEvent fe) {

if (!iskeyAvilable()) {

searchBar.setText("Search...");

initHome();

}

}

});

sortOption.addItem("Relevance");

sortOption.addItem("Price: Low to High");

sortOption.addItem("Price: High to Low");

sortOption.addItem("Product Name: A - Z");

sortOption.addItem("Product Name: Z - A");

sortOption.addItem("Category: A - Z");

sortOption.addItem("Category: Z - A");

sortOption.addItemListener(new ItemListener() {

public void itemStateChanged(ItemEvent ie) {

if (ie.getStateChange() == ItemEvent.SELECTED) {

updateList();

if (shopItems.isEmpty())

emptyMSG.setText(FILTER\_MSG);

}

}

});

for (Container c : new Container[]{back, reset, action, bNext, featuredPanel, featuredLabel})

c.setVisible(false);

panel.setOpaque(false);

panel.getViewport().setOpaque(false);

shopItemList.setListData(shopItems);

shopItemList.setCellRenderer(new ListItem(getCodeBase(), false));

shopItemList.setOpaque(false);

shopItemList.addMouseListener(l);

shopItemList.addListSelectionListener(new ListSelectionListener() {

public void valueChanged(ListSelectionEvent lse) {

int[] is = shopItemList.getSelectedIndices();

action.setVisible(is.length != 0);

}

});

shopItemList.addKeyListener(new KeyListener() {

public void keyTyped(KeyEvent ke) {}

public void keyPressed(KeyEvent ke) {}

public void keyReleased(KeyEvent ke) {

if (ke.getKeyChar() == KeyEvent.VK\_ENTER)

actionPerformed(new ActionEvent(action, 0, null));

}

});

Container c = new Container();

c.setLayout(new BorderLayout(7, 7));

c.add(next, BorderLayout.EAST);

c.add(action, BorderLayout.CENTER);

navBar.setLayout(new BorderLayout(7, 7));

navBar.add(back, BorderLayout.WEST);

navBar.add(searchBar, BorderLayout.CENTER);

navBar.add(c, BorderLayout.EAST);

bottomBar.setLayout(new BorderLayout(7, 7));

content.setLayout(new BorderLayout(7, 7));

JLabel l1 = new JLabel("Sort: ");

l1.setForeground(Color.decode("#BDBDBD"));

JLabel l2 = new JLabel("Shop by Category:");

l2.setForeground(Color.decode("#BDBDBD"));

Container opts = new Container();

opts.setLayout(new GridLayout(0, 1, 0, 5));

opts.add(l1);

opts.add(sortOption);

opts.add(l2);

for (int i = 0; i < catOptions.length; i++) {

catOptions[i] = new JCheckBox(Category.values()[i].name());

catOptions[i].addItemListener(new ItemListener() {

public void itemStateChanged(ItemEvent ie) {

updateList();

if (shopItems.isEmpty())

emptyMSG.setText(FILTER\_MSG);

}

});

catOptions[i].setOpaque(false);

catOptions[i].setForeground(Color.WHITE);

opts.add(catOptions[i]);

}

opts.add(reset);

Container sideOptions = new Container();

sideOptions.setLayout(new BorderLayout(7, 7));

sideOptions.add(opts, BorderLayout.NORTH);

sideMenu.setLayout(new BorderLayout(7, 7));

sideMenu.add(sideTitle, BorderLayout.NORTH);

sideMenu.add(sideOptions, BorderLayout.CENTER);

sideMenu.setVisible(false);

emptyMSG.setFont(new Font(null, Font.BOLD, 20));

emptyMSG.setForeground(Color.WHITE);

emptyMSG.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

featuredPanel.setOpaque(false);

featuredPanel.getViewport().setOpaque(false);

featured.setLayout(new BorderLayout(7, 7));

featured.add(featuredLabel, BorderLayout.NORTH);

featured.add(featuredPanel, BorderLayout.CENTER);

featured.add(appLabel, BorderLayout.SOUTH);

featuredItemsList.setLayoutOrientation(JList.VERTICAL\_WRAP);

featuredItemsList.setFixedCellWidth(screenSize.width / 3 - 30);

featuredItemsList.setVisibleRowCount(2);

featuredItemsList.setCellRenderer(new ListItem(getCodeBase(), true));

featuredItemsList.setOpaque(false);

featuredItemsList.addMouseListener(l);

featuredItemsList.addListSelectionListener(new ListSelectionListener() {

public void valueChanged(ListSelectionEvent lse) {

int[] is = featuredItemsList.getSelectedIndices();

action.setVisible(is.length != 0);

}

});

featuredItemsList.addKeyListener(new KeyListener() {

public void keyTyped(KeyEvent ke) { }

public void keyPressed(KeyEvent ke) {}

public void keyReleased(KeyEvent ke) {

if (ke.getKeyChar() == KeyEvent.VK\_ENTER)

actionPerformed(new ActionEvent(action, 0, null));

}

});

mainContainer.setLayout(new BorderLayout(7, 7));

mainContainer.add(featured, BorderLayout.NORTH);

mainContainer.add(panel, BorderLayout.CENTER);

content.add(sideMenu, BorderLayout.WEST);

content.add(mainContainer, BorderLayout.CENTER);

standardButton.setOpaque(false);

standardButton.setForeground(Color.WHITE);

standardButton.setFont(new Font(null, Font.BOLD, 14));

standardButton.setSelected(true);

expressButton.setOpaque(false);

expressButton.setForeground(Color.WHITE);

expressButton.setFont(new Font(null, Font.BOLD, 14));

standardButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

extra = 0;

summary.setText("Total: " + total);

expressButton.setSelected(false);

}

});

expressButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

extra = 250;

summary.setText("Total: " + total + " +" + extra);

standardButton.setSelected(false);

}

});

JPanel jp = new JPanel();

jp.setPreferredSize(new Dimension(540, 125));

jp.setBorder(BorderFactory.createLineBorder(Color.WHITE, 3));

jp.setOpaque(false);

SpringLayout layout1 = new SpringLayout();

layout1.putConstraint(SpringLayout.HORIZONTAL\_CENTER, jp,

0, SpringLayout.HORIZONTAL\_CENTER, deliveryContainer);

layout1.putConstraint(SpringLayout.VERTICAL\_CENTER, jp,

0, SpringLayout.VERTICAL\_CENTER, deliveryContainer);

layout1.putConstraint(SpringLayout.HORIZONTAL\_CENTER, standardButton,

0, SpringLayout.HORIZONTAL\_CENTER, deliveryContainer);

layout1.putConstraint(SpringLayout.VERTICAL\_CENTER, standardButton,

-30, SpringLayout.VERTICAL\_CENTER, deliveryContainer);

layout1.putConstraint(SpringLayout.HORIZONTAL\_CENTER, expressButton,

0, SpringLayout.HORIZONTAL\_CENTER, deliveryContainer);

layout1.putConstraint(SpringLayout.VERTICAL\_CENTER, expressButton,

30, SpringLayout.VERTICAL\_CENTER, deliveryContainer);

deliveryContainer.setLayout(layout1);

deliveryContainer.add(standardButton);

deliveryContainer.add(jp);

deliveryContainer.add(expressButton);

SpringLayout layout2 = new SpringLayout();

JPanel panel2 = new JPanel();

panel2.setPreferredSize(new Dimension(440, 150));

panel2.setBorder(BorderFactory.createLineBorder(Color.WHITE, 3));

panel2.setOpaque(false);

ccInfo.setForeground(Color.WHITE);

ccInfo.setPreferredSize(new Dimension(200, 40));

sccNumber.setForeground(Color.WHITE);

ccNumber.setForeground(Color.WHITE);

ccNumber.setPreferredSize(new Dimension(200, 40));

enterCC.setForeground(Color.WHITE);

layout2.putConstraint(SpringLayout.HORIZONTAL\_CENTER, panel2,

0, SpringLayout.HORIZONTAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.VERTICAL\_CENTER, panel2,

-10, SpringLayout.VERTICAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.HORIZONTAL\_CENTER, ccInfo,

30, SpringLayout.HORIZONTAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.VERTICAL\_CENTER, ccInfo,

-60, SpringLayout.VERTICAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.HORIZONTAL\_CENTER, sccNumber,

-100, SpringLayout.HORIZONTAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.VERTICAL\_CENTER, sccNumber,

-20, SpringLayout.VERTICAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.VERTICAL\_CENTER, enterCC,

20, SpringLayout.VERTICAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.HORIZONTAL\_CENTER, ccNumber,

100, SpringLayout.HORIZONTAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.VERTICAL\_CENTER, ccNumber,

-20, SpringLayout.VERTICAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.HORIZONTAL\_CENTER, enterCC,

-70, SpringLayout.HORIZONTAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.HORIZONTAL\_CENTER, cvvField,

20, SpringLayout.HORIZONTAL\_CENTER, paymentConatiner);

layout2.putConstraint(SpringLayout.VERTICAL\_CENTER, cvvField,

20, SpringLayout.VERTICAL\_CENTER, paymentConatiner);

paymentConatiner.setLayout(layout2);

for (Container l : new Container[]{ccInfo, cvvField, ccNumber, sccNumber, enterCC, panel2}) {

paymentConatiner.add(l);

if (l instanceof JLabel)

((JLabel) l).setFont(new Font(null, Font.BOLD, 14));

}

Container c1 = new Container();

c1.setLayout(new BorderLayout(7, 7));

c1.add(summary, BorderLayout.EAST);

bottomBar.add(c1, BorderLayout.CENTER);

bottomBar.add(bNext, BorderLayout.EAST);

bottomBar.setVisible(false);

shop.setLayout(new BorderLayout(7, 7));

shop.add(navBar, BorderLayout.NORTH);

shop.add(content, BorderLayout.CENTER);

shop.add(bottomBar, BorderLayout.SOUTH);

setLayout(new CardLayout(7, 7));

add(shop);

moveToCenter();

}

public void paint(Graphics graphics) {

Image image = getImage(getCodeBase(), "background.jpg");

graphics.drawImage(image, 0, 0, (int) getBounds().getWidth(), (int) getBounds().getHeight(), this);

super.paint(graphics);

}

public void actionPerformed(ActionEvent e) {

Object o = e.getSource();

if (o == next || o == bNext)

switch (page) {

case HOME:

initCart();

break;

case SEARCH:

initHome();

initCart();

break;

case CART:

initDelivery();

break;

case DELIVERY\_OPTIONS:

initPayment();

break;

case PAYMENT:

JOptionPane optionPane;

if (cvvField.getText().equals(CVV)) {

optionPane = new JOptionPane("Thank you for purching", JOptionPane.PLAIN\_MESSAGE,

JOptionPane.CLOSED\_OPTION);

db.removeAll(cartItems);

purchaseItems.addAll(cartItems);

featuredItems.addAll(cartItems);

cartItems.clear();

cvvField.setText("");

initHome();

} else {

optionPane = new JOptionPane("Invalid CVV", JOptionPane.WARNING\_MESSAGE,

JOptionPane.DEFAULT\_OPTION);

}

final JDialog dialog = new JDialog((Frame) null, "Message", true);

optionPane.addPropertyChangeListener(new PropertyChangeListener() {

public void propertyChange(PropertyChangeEvent e) {

if (dialog.isVisible() && e.getPropertyName().equals(JOptionPane.VALUE\_PROPERTY)) {

dialog.setVisible(false);

}

}

});

dialog.setContentPane(optionPane);

dialog.pack();

dialog.setLocationRelativeTo(null);

dialog.setVisible(true);

break;

}

else if (o == back)

switch (page) {

case CART:

case SEARCH:

initHome();

break;

case PAYMENT:

initDelivery();

break;

case DELIVERY\_OPTIONS:

initCart();

break;

}

else if (o == reset) {

resetFilter();

updateList();

} else if (o.equals(action)) {

int[] is = shopItemList.getSelectedIndices(), i1s = featuredItemsList.getSelectedIndices();

if (is.length != 0 || i1s.length != 0) {

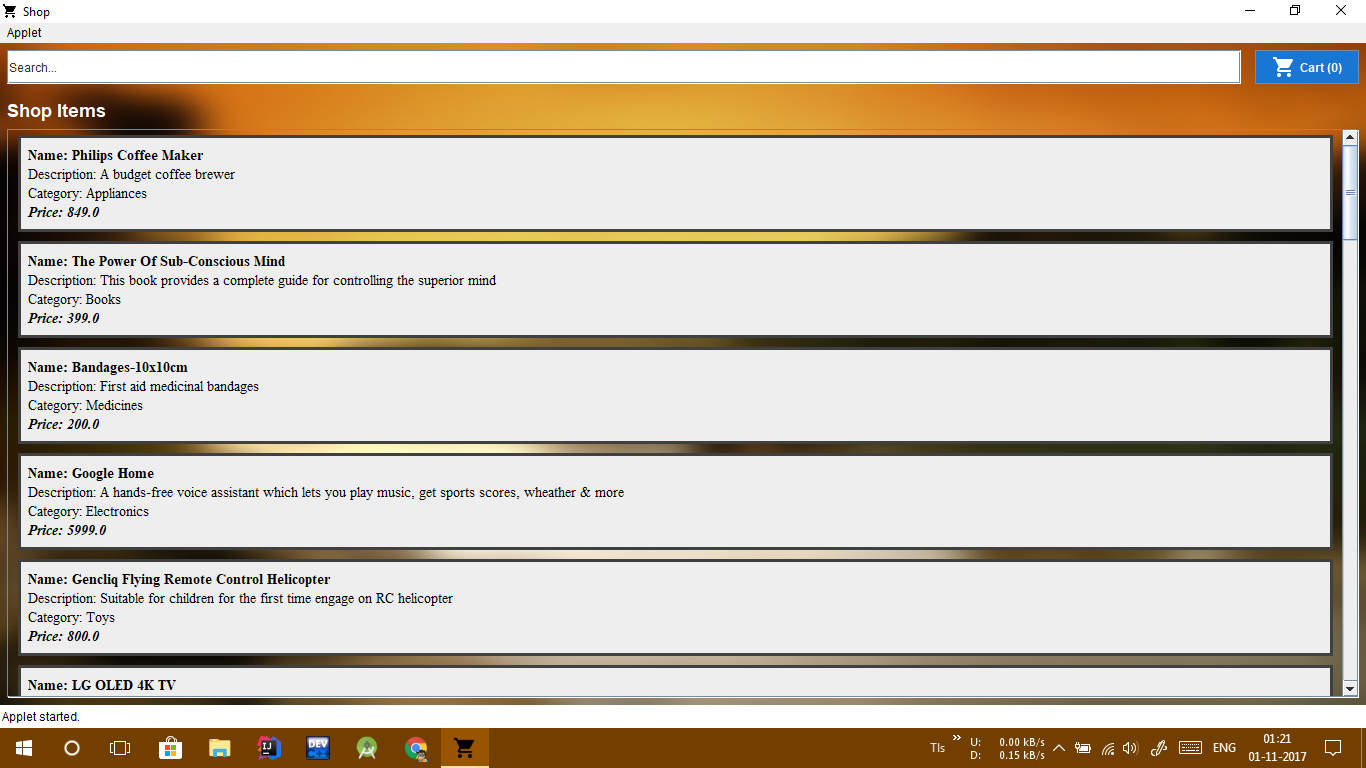
ProductList pl = new ProductList();

switch (page) {

case SEARCH:

case HOME:

for (int i : is) pl.add(shopItems.get(i));

for (int i : i1s) pl.add(featuredItems.get(i));

cartItems.addAll(pl);

db.removeAll(pl);

featuredItems.removeAll(pl);

shopItems.removeAll(pl);

next.setText("Cart (" + cartItems.size() + ")");

shopItemList.setListData(shopItems);

if (page == HOME) {

if (shopItems.isEmpty()) {

emptyMSG.setText(HOME\_MSG);

shop.remove(mainContainer);

shop.add(emptyMSG, BorderLayout.CENTER);

} else {

shop.add(mainContainer, BorderLayout.CENTER);

shop.remove(emptyMSG);

}

boolean b = featuredItems.isEmpty();

featuredLabel.setVisible(!b);

featuredPanel.setVisible(!b);

}

break;

case CART:

for (int i : is) pl.add(cartItems.get(i));

cartItems.removeAll(pl);

db.addAll(pl);

shopItems.addAll(pl);

next.setText("Check out (" + cartItems.size() + ")");

shopItemList.setListData(cartItems);

if (cartItems.isEmpty()) {

emptyMSG.setText(CART\_MSG);

next.setVisible(false);

shop.remove(mainContainer);

shop.add(emptyMSG, BorderLayout.CENTER);

summary.setVisible(false);

} else {

total = 0;

for (Product p : cartItems) total += p.price;

next.setVisible(true);

summary.setText("Total: " + total);

summary.setVisible(true);

shop.add(mainContainer, BorderLayout.CENTER);

shop.remove(emptyMSG);

}

break;

}

repaint();

action.setVisible(false);

}

}

}

private void resetFilter() {

for (JCheckBox box : catOptions) box.setSelected(false);

sortOption.setSelectedIndex(0);

}

private void filter() {

shopItems = new ProductList(db);

int sI = sortOption.getSelectedIndex();

selectedCat.clear();

for (int i = 0; i < catOptions.length; i++)

if (catOptions[i].isSelected()) selectedCat.add(i);

if (sI == 0 && !purchaseItems.isEmpty() && page == SEARCH)

shopItems = new ProductList(db.getSimilar(purchaseItems, db.size()));

reset.setVisible(sI != 0 || !selectedCat.isEmpty());

if (!reset.isVisible()) return;

if (!selectedCat.isEmpty()) {

shopItems.clear();

for (int i : selectedCat)

for (Category c : Category.values())

if (i == c.ordinal()) {

ProductList temp = new ProductList(db);

shopItems.addAll(temp.filter(c));

break;

}

}

switch (sI) {

case 1:

case 2:

shopItems.sortByPrice(sI == 2);

break;

case 3:

case 4:

shopItems.sortByName(sI == 4);

break;

case 5:

case 6:

shopItems.sortByCategory(sI == 6);

break;

}

}

private void updateList() {

filter();

if (!iskeyAvilable()) back.setVisible(false);

else {

page = SEARCH;

shopItems = shopItems.search(key);

back.setVisible(true);

}

shopItemList.setListData(shopItems);

if (shopItems.isEmpty()) {

content.remove(mainContainer);

content.add(emptyMSG, BorderLayout.CENTER);

} else {

content.remove(emptyMSG);

content.add(mainContainer, BorderLayout.CENTER);

}

repaint();

}

private boolean iskeyAvilable() {

return !key.isEmpty();

}

private void moveToCenter() {

Container c = getParent();

Frame frame = null;

int i = 0;

while (c.getParent() != null) c = c.getParent();

if (c instanceof Frame) frame = (Frame) c;

if (frame != null) {

frame.setLocationRelativeTo(null);

frame.setTitle("Shop");

frame.setIconImage(getImage(getCodeBase(), "logo.png"));

}

}

private void initHome() {

next.setIcon(cartIcon);

next.setText("Cart (" + cartItems.size() + ")");

action.setIcon(addIcon);

action.setToolTipText("Add to Cart");

appLabel.setText("Shop Items");

switch (page) {

case CART:

shopItemList.setListData(shopItems);

bottomBar.setVisible(false);

if (cartItems.isEmpty()) {

next.setVisible(true);

shop.remove(emptyMSG);

shop.add(mainContainer, BorderLayout.CENTER);

}

break;

case SEARCH:

searchBar.setText("Search...");

key = "";

featured.setVisible(true);

sideMenu.setVisible(false);

resetFilter();

shopItems.shuffle();

updateList();

break;

case PAYMENT:

featuredItemsList.setListData(featuredItems);

shopItemList.setListData(shopItems);

for (Container c : new Container[]{next, mainContainer, bottomBar})

c.setVisible(c != bottomBar);

shop.remove(paymentConatiner);

navBar.remove(title);

navBar.add(searchBar, BorderLayout.CENTER);

updateList();

featuredItems.clear();

featuredItems.addAll(db.getSimilar(purchaseItems, 5));

shopItems.removeAll(featuredItems);

break;

}

if (shopItems.isEmpty()) {

emptyMSG.setText(HOME\_MSG);

shop.remove(panel);

shop.add(emptyMSG, BorderLayout.CENTER);

if (featuredItems.isEmpty()) {

shop.removeAll();

return;

}

}

boolean b = !featuredItems.isEmpty();

featuredLabel.setVisible(b);

featuredPanel.setVisible(b);

back.setVisible(false);

page = HOME;

repaint();

}

private void initCart() {

switch (page) {

case HOME:

action.setToolTipText("Remove from Cart");

back.setVisible(true);

next.setText("Check out (" + cartItems.size() + ")");

next.setIcon(checkOutIcon);

shopItemList.setListData(cartItems);

action.setIcon(removeIcon);

featuredLabel.setVisible(false);

featuredPanel.setVisible(false);

appLabel.setText("Cart Items");

if (cartItems.isEmpty()) {

next.setVisible(false);

emptyMSG.setText(CART\_MSG);

shop.remove(mainContainer);

shop.add(emptyMSG, BorderLayout.CENTER);

} else {

total = 0;

for (Product p : cartItems) total += p.price;

for (Container c : new Container[]{bottomBar, next, summary})

c.setVisible(true);

summary.setText("Total: " + total);

shop.add(mainContainer, BorderLayout.CENTER);

shop.remove(emptyMSG);

}

break;

case DELIVERY\_OPTIONS:

if (shopItemList.getSelectedIndices().length > 0) action.setVisible(true);

for (Container c : new Container[]{bNext, next, mainContainer})

c.setVisible(c != bNext);

shop.remove(deliveryContainer);

navBar.add(searchBar, BorderLayout.CENTER);

navBar.remove(title);

}

page = CART;

bNext.setVisible(false);

repaint();

}

private void initDelivery() {

switch (page) {

case CART:

for (Container c : new Container[]{action, next, mainContainer})

c.setVisible(false);

navBar.remove(searchBar);

navBar.add(title, BorderLayout.CENTER);

bNext.setVisible(true);

break;

case PAYMENT:

shop.remove(paymentConatiner);

break;

}

bNext.setIcon(payIcon);

bNext.setText("Proceed To Payment");

title.setText("Delivery Options");

shop.add(deliveryContainer, BorderLayout.CENTER);

page = DELIVERY\_OPTIONS;

repaint();

}

private void initPayment() {

page = PAYMENT;

summary.setText("Total: " + (total + extra));

bNext.setText("Pay");

title.setText("Payment");

shop.remove(deliveryContainer);

shop.add(paymentConatiner, BorderLayout.CENTER);

repaint();

}

}

Class ListItem

package ui;

import java.awt.\*;

import java.net.\*;

import javax.swing.\*;

import javax.swing.border.\*;

import product.Product;

public class ListItem extends JEditorPane implements ListCellRenderer<Product> {

private boolean isSelected;

private boolean onfeatured;

private URL codeBase;

private Product product;

public ListItem(URL codeBase, boolean b) {

super();

this.onfeatured = b;

this.codeBase = codeBase;

}

public void paint(Graphics g) {

g.setColor(isSelected ? Color.WHITE : Color.decode("#eeeeee"));

g.fillRect(10, 5, getWidth() - 20, getHeight() - 10);

g.setColor(isSelected ? Color.BLUE : Color.DARK\_GRAY);

drawBorder(g, 3);

super.paint(g);

}

private void drawBorder(Graphics g, int t) {

for (int i = isSelected ? 9 : 10, j = isSelected ? 4 : 5, k = 0; k < t; i++, j++, k++)

g.drawRect(i, j, getWidth() - 2 \* i, getHeight() - 2 \* j);

}

public Container getListCellRendererComponent(JList<? extends Product> jlist, Product product, int index, boolean isSelected, boolean cellHasFocus) {

setContentType("text/html");

product.onfeatured(onfeatured);

this.isSelected = isSelected;

this.product = product;

setText(product.getHtmlString());

if (onfeatured) setToolTipText(product.description);

setEditable(false);

setOpaque(false);

setFont(new Font(null, Font.BOLD, 14));

Border paddingBorder = new EmptyBorder(15, 20, 15, 20);

setBorder(paddingBorder);

return this;

}

}

Class Utils

package utilites;

import java.util.\*;

import lib.RandomCollection;

import product.\*;

import product.ProductList.Category;

public class Utils {

public static final int PRICE\_MODE = 1;

public static final int NAME\_MODE = 2;

public static final int CAT\_MODE = 3;

public static void sort(ProductList list, int mode, boolean dec) {

int n = list.size();

if (mode == CAT\_MODE) sort(list, NAME\_MODE, dec);

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1 - i; j++) {

Product pj = list.get(j), pj1 = list.get(j + 1);

switch (mode) {

case NAME\_MODE:

if (pj.name.compareTo(pj1.name) > 0) list.swap(j, j + 1);

break;

case PRICE\_MODE:

if (pj.price > pj1.price) list.swap(j, j + 1);

break;

case CAT\_MODE:

if (pj.category.ordinal() > pj1.category.ordinal()) list.swap(j, j + 1);

break;

}

}

}

if (dec) for (int i = 0; i < n / 2; i++) list.swap(i, n - i - 1);

}

private final static String[] COMMON\_TERMS = new String[]{

"in", "with", "on", "by", "under", "a", "the",

"an", "then", "that", "when", "also", "as",

"get", "you", "will", "used", "gain", "can" };

public static ProductList filter(ProductList list, Category c) {

for (Product p : list.toArray(new Product[list.size()]))

if (!p.category.equals(c)) list.remove(p);

return list;

}

public static ProductList search(ProductList list, String s) {

ProductList result = new ProductList();

if (s.isEmpty()) return result;

String in = s.toLowerCase();

String a[] = in.split(" ");

Vector<String> keys = new Vector<>();

keys.addAll(Arrays.asList(a));

keys.removeAll(Arrays.asList(COMMON\_TERMS));

for (Product p : list)

for (String key : keys) {

String c = p.name + p.description + p.price + p.category.toString();

c = c.toLowerCase();

if (c.contains(key) && result.indexOf(p) == -1) result.add(p);

}

if (in.contains("under")) {

if (result.isEmpty()) result = new ProductList(list);

long num = 0L;

String number = "";

char[] chars = in.toCharArray();

for (int i = in.indexOf("under") + 5; i < in.length(); i++) {

char c = chars[i];

if (Character.isDigit(c)) {

while (i < in.length() && Character.isDigit(c = chars[i++])) number += c;

break;

}

}

if (!number.isEmpty()) num = Long.parseLong(number);

if (num != 0) for (Product p : result.toArray(new Product[result.size()]))

if (p.price > num) result.remove(p);

}

return result;

}

public static ProductList getSimilarProducts(ProductList db, ProductList list, int length) {

ProductList similar = new ProductList(), ref = new ProductList(db);

float prop[] = new float[Category.values().length];

RandomCollection<Category> collection = new RandomCollection<>();

Arrays.fill(prop, .1f);

if (length == db.size()) Arrays.fill(prop, prop.length);

for (Product p : list) prop[p.category.ordinal()]++;

System.out.println("Frequency: " + Arrays.toString(prop));

for (int i = 0; i < prop.length; i++) {

prop[i] /= list.size();

collection.add(prop[i], Category.values()[i]);

}

System.out.println("Probability: " + Arrays.toString(prop));

ref.sortByCategory(false);

while (!ref.isEmpty() && similar.size() < length) {

Category c = collection.next();

for (Product p : ref)

if (p.category == c) {

similar.add(p);

ref.remove(p);

break;

}

}

return similar;

}

}

Class Product

package product;

import product.ProductList.Category;

public class Product {

public String name, description;

public double price;

public Category category;

public String image;

private boolean onfeatured;

public Product(String a, String b, double c, Category d) {

name = a;

description = b;

price = c;

category = d;

}

public String getHtmlString() {

return toString().replace("\n", "<br>").replace("%b", "<strong>").replace("%/b", "</strong>").replace("%i", "<i>").replace("%/p ", "</i>");

}

public String toString() {

return "%bName: " + name + (onfeatured ? "%/b" : "%/b\nDescription: " + description) + "\nCategory: " + category + "\n%i%bPrice: " + price + "%/b%i\n";

}

public void onfeatured(boolean onfeatured) {

this.onfeatured = onfeatured;

}

}

Class ProductList

package product;

import java.util.\*;

import utilites.Utils;

public class ProductList extends Vector<Product> {

public ProductList() {

super();

}

public ProductList(ProductList list) {

super(list);

}

public ProductList filter(Category category) {

return Utils.filter(this, category);

}

public enum Category {

Appliances, Books, Electronics, Fashion, Grocery, Medicines, Toys

}

public static ProductList getDatabase() {

ProductList all = new ProductList();

all.add(new Product("Google Pixel 2",

"A android smartphone by google",

61000, Category.Electronics));

all.add(new Product("Schindler's List",

"Book about a greedy German businessman",

1000, Category.Books));

all.add(new Product("Whirlpool Single Door Refrigerators",

"A SmartFridge with 3 star BEE energy rating",

10490, Category.Appliances));

all.add(new Product("Voltas 1.4 Ton 3 Star Split AC",

"A split ac with Sleep mode and Turbo mode ",

27990, Category.Appliances));

all.add(new Product("HUL Pureit Advanced RO+UV 5-Litre Water Purifier",

"The Pureit purifier guarantees that you will get as safe as boiled water",

8399, Category.Appliances));

all.add(new Product("Saffire Kids Twist Scooter",

"Children can get moving on the Saffire Twist Scooter and gain confidence with speed and balance",

1299, Category.Toys));

all.add(new Product("Gencliq Flying Remote Control Helicopter",

"Suitable for children for the first time engage on RC helicopter",

800, Category.Toys));

all.add(new Product("Google Home",

"A hands-free voice assistant which lets you play music, get sports scores, wheather & more",

5999, Category.Electronics));

all.add(new Product("Asus VivoBook Laptop ",

"A powerful laptop with an Intel 7th Gen Processor, 8GB RAM, Nvidia GTX 1050 Ti & a stunning 1080p display",

47999, Category.Electronics));

all.add(new Product("Catcher in the Rye",

"The Catcher in the Rye is a 1951 novel by J. D. Salinger describing a story of a man undegoing treatment in a mental hospital",

499, Category.Books));

all.add(new Product("FastTrack Men's Watch",

"High quality leather buckle with a black quartz dial",

2399, Category.Fashion));

all.add(new Product("Cadbury Celebrations- Small Pack",

"A sweet gift for a sweet ocassion",

399, Category.Grocery));

all.add(new Product("Miranda 600ml Pack of 50",

"Soft-Drink with no artificial flavour",

1199, Category.Grocery));

all.add(new Product("Bandages-10x10cm",

"First aid medicinal bandages",

200, Category.Medicines));

all.add(new Product("First Aid Travel Kit ",

"A light & compact first aid kit which consists of essentials supplies like bandages,cotton buds, sterile guaze, alcohol wipes etc",

999, Category.Medicines));

all.add(new Product("LG OLED 4K TV",

"A wifi enabled smart TV with 60 Hz refresh rate & an incredible sound system",

89499, Category.Electronics));

all.add(new Product("Sandisk 16GB Pendrive",

"Portable & a high speed storage device",

349, Category.Electronics));

all.add(new Product("Sennheiser HD 180 Over-Ear Headphones (Black)",

"Over the head earphones with superior sound quality",

990, Category.Electronics));

all.add(new Product("HP DeskJet Ink Advantage 2135 All-in-One Printer",

"Copy, scan & print through a single compact device",

4790, Category.Electronics));

all.add(new Product("iPad Mini 2 Tablet",

"The smallest & the fastest tablet with a powerful A8X chip & touch-ID",

34990, Category.Electronics));

all.add(new Product("Core Python Programming",

"The complete reference for Python",

599, Category.Books));

all.add(new Product("Da-Vinci Code",

"A book about a mysterious & a bizzare murder",

599, Category.Books));

all.add(new Product("The Power Of Sub-Conscious Mind",

"This book provides a complete guide for controlling the superior mind",

399, Category.Books));

all.add(new Product("To Kill a Mockingbird",

"An all-time best seller about growing up in extraordinary circumstances",

699, Category.Books));

all.add(new Product("The Martian",

"A fiction based novel about an astronaut",

449, Category.Books));

all.add(new Product("Philips Coffee Maker",

"A budget coffee brewer",

849, Category.Appliances));

all.add(new Product("Godrej Fully Automatic Washing Machine",

"Top load easy to use washing machine",

14449, Category.Appliances));

all.add(new Product("Bajaj 1000 Watt Iron",

"A plug & play appliance",

1449, Category.Appliances));

all.add(new Product("LG Microwave",

"A microwave with 30 liters of capacity & upto 20 auto cooking options",

6449, Category.Appliances));

all.add(new Product("Solimo Permium Almonds 500gm",

"High Quality Dry Fruits",

449, Category.Appliances));

all.shuffle();

return all;

}

public void sortByName(boolean dec) {

Utils.sort(this, Utils.NAME\_MODE, dec);

}

public void sortByPrice(boolean dec) {

Utils.sort(this, Utils.PRICE\_MODE, dec);

}

public void sortByCategory(boolean dec) {

Utils.sort(this, Utils.CAT\_MODE, dec);

}

public ProductList getSimilar(ProductList list, int length) {

return Utils.getSimilarProducts(this, list, length);

}

public ProductList search(String s) {

return Utils.search(this, s);

}

public void shuffle() {

Collections.shuffle(this);

}

public void swap(int a, int b) {

Product pa = get(a), pb = get(b);

remove(pa); remove(pb);

add(a, pb); add(b, pa);

}

}

Output:

