package myProject;

//Easy GUI Apps, Components, IBIO

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.\*;

//=========================================================================

//==== EasyApp - standard GUI app with standard IBIO routines ============

//=========================================================================

public class EasyApp extends Frame implements ActionListener, ItemListener, MouseListener, KeyListener

{

//--- Constructor -------

MenuBar menubar = addMenuBar();

public EasyApp()

{

setLayout(null);

setSize(600,400);

addWindowListener // enable [x] for closing app

( new WindowAdapter()

{ public void windowClosing(WindowEvent evt)

{ evt.getWindow().dispose(); }

}

);

setVisible(true);

toFront();

requestFocus();

}

public MenuBar makeMenus(String[] menus)

{

for (int x=0;x<menus.length;x++)

{ if (menus[x].length()>0)

{ int d = menus[x].indexOf("|");

if (d<0) {d=menus[x].length(); }

Menu newmenu = addMenu(menus[x].substring(0,d),menubar);

while (d>=0 && d<menus[x].length())

{

int e = menus[x].indexOf("|",d+1);

if (e<0) {e=menus[x].length(); }

MenuItem newmi = addMenuItem(menus[x].substring(d+1,e),newmenu,this);

newmi.setActionCommand(newmenu.getActionCommand()+"|"+newmi.getActionCommand());

d = e;

}

}

}

return menubar;

}

//============================

//--- Control Creators -------

//============================

public Button addButton(String text, int left, int top, int width, int height,

ActionListener listener)

{

Button newComp = new Button(text);

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addActionListener(listener);

return newComp;

}

public Checkbox addCheckbox(String text, int left, int top, int width, int height,

ItemListener listener)

{

Checkbox newComp = new Checkbox(text);

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addItemListener(listener);

return newComp;

}

public TextField addTextField(String text, int left, int top, int width, int height,

ActionListener listener)

{

TextField newComp = new TextField(text);

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addActionListener(listener);

return newComp;

}

public TextArea addTextArea(String text, int left, int top, int width, int height,

KeyListener listener)

{

TextArea newComp = new TextArea(text);

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addKeyListener(listener);

return newComp;

}

public Label addLabel(String text, int left, int top, int width, int height,

MouseListener listener)

{

Label newComp = new Label(text);

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addMouseListener(listener);

return newComp;

}

public JLabel addJLabel(ImageIcon img, int left, int top, int width, int height,

MouseListener listener)

{

JLabel newComp = new JLabel(img);

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addMouseListener(listener);

return newComp;

}

public List addList(String words,int left, int top, int width, int height,

ItemListener listener)

{

List newComp = new List();

while (words.length()>0)

{ int s = words.indexOf("|");

if (s<0)

{ newComp.add(words);

words = "";

}

else

{ newComp.add(words.substring(0,s));

words = words.substring(s+1);

}

}

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addItemListener(listener);

return newComp;

}

public Choice addChoice(String words,int left, int top, int width, int height,

ItemListener listener)

{

Choice newComp = new Choice();

while (words.length()>0)

{ int s = words.indexOf("|");

if (s<0)

{ newComp.add(words);

words = "";

}

else

{ newComp.add(words.substring(0,s));

words = words.substring(s+1);

}

}

add(newComp);

newComp.setBounds(left,top,width,height);

if (listener!=null)newComp.addItemListener(listener);

return newComp;

}

public JTable addJTable(String[][] data, int left, int top, int width, int height,

ActionListener listener)

{

String[] heads = new String[data[0].length];

for(int s=0; s<heads.length; s++){ heads[s] = ""; }

JTable newComp = new JTable(data , heads);

JScrollPane newComps = new JScrollPane(newComp);

add(newComps);

newComps.setBounds(left,top,width,height);

// if (listener!=null)newComp.addActionListener(listener);

return newComp;

}

public MenuItem addMenuItem(String mText,Menu mu,ActionListener listener)

{ MenuItem c = new MenuItem(mText);

mu.add(c);

if (listener!=null)c.addActionListener(listener);

return c;

}

public Menu addMenu(String mText,Menu mu)

{

Menu c = new Menu(mText);

mu.add(c);

return c;

}

public Menu addMenu(String mText,MenuBar mb)

{

Menu c = new Menu(mText);

mb.add(c);

return c;

}

public Menu addMenu(String mText)

{

int d = mText.indexOf("|");

if (d<0){d=mText.length();}

String id = mText.substring(0,d);

Menu c = new Menu(id);

menubar.add(c);

while (d<mText.length())

{ int e = mText.indexOf("|",d+1);

if (e<0){e = mText.length();}

String cmd = mText.substring(d+1,e);

MenuItem mi = addMenuItem(cmd,c,this);

mi.setActionCommand(id+"|"+cmd);

c.add(mi);

d = e;

}

return c;

}

public MenuBar addMenuBar()

{ MenuBar c = new MenuBar();

setMenuBar(c);

return c;

}

//==================================

//=== Simplified Command Library ===

//==================================

public Process runProgram(String programName)

{

Process handle = null;

try

{ handle = Runtime.getRuntime().exec(programName);

}

catch(IOException ex)

{ };

return handle;

}

public String chooseFile()

{ FileDialog fd = new FileDialog(this);

fd.setVisible(true);

return fd.getDirectory() + fd.getFile();

}

public Image loadImage(String filename)

{ ImageIcon im = new ImageIcon(filename);

int c = 0;

do

{ c++;

} while (c<100000 && im.getImageLoadStatus()!=MediaTracker.COMPLETE);

return im.getImage();

}

public boolean loadList(List list,String filename)

{

try

{ list.removeAll();

BufferedReader infile = new BufferedReader(new FileReader(filename));

while(infile.ready())

{ list.add(infile.readLine());

}

infile.close();

return true;

}

catch(IOException e)

{ output(e.toString());

return false;

}

}

public boolean saveList(String filename,List list)

{

try

{ PrintWriter outfo = new PrintWriter(new FileWriter(filename));

for (int c=0;c<list.getItemCount();c++)

{ outfo.println(list.getItem(c));

}

outfo.close();

return true;

}

catch(IOException e)

{ output(e.toString());

return false;

}

}

public boolean loadArray(String[] array,String filename)

{

try

{

BufferedReader countFile = new BufferedReader(new FileReader(filename));

int s = 0;

while(countFile.ready())

{ array[s] = countFile.readLine();

s++;

}

array[s] = null;

countFile.close();

return true;

}

catch(Exception e)

{ output(e.toString());

return false;

}

}

public boolean saveArray(String filename,String[] array)

{

try

{ PrintWriter outfo = new PrintWriter(new FileWriter(filename));

int c = 0;

while (array[c]!=null)

{ outfo.println(array[c]);

c++;

}

outfo.close();

return true;

}

catch(IOException e)

{ output(e.toString());

return false;

}

}

public void sortArray(String[] arr)

{

int swaps = 0 ;

do

{ swaps = 0;

int c = 0;

while (arr[c+1]!=null)

{ if (arr[c].compareTo(arr[c+1])>0)

{ String temp = arr[c];

arr[c] = arr[c+1];

arr[c+1] = temp;

swaps++;

}

c++;

}

} while (swaps != 0);

}

public String getType(Object ob)

{

String c = ob.getClass().toString();

int p = c.lastIndexOf(".");

if (p<0) {p=-1;}

return c.substring(p+1);

}

//=== Stub versions of Listeners to satisfy interfaces ===

public void actionPerformed(ActionEvent evt)

{ actions(evt.getSource(),evt.getActionCommand());

}

public void itemStateChanged(ItemEvent evt)

{ Object source = evt.getSource();

String t = getType(source);

String command = "";

if (t.equals("List")) { command = ((List)source).getSelectedItem(); }

else if (t.equals("Choice")){ command = ((Choice)source).getSelectedItem(); }

else {command = evt.getItem().toString();}

actions(source,command);

}

public void actions(Object source,String command)

{

output(source.toString()+":"+command);

}

public void mouseClicked(MouseEvent e)

{ Object source = e.getSource();

actions(source,"");

}

public void mouseEntered(MouseEvent e){}

public void mouseExited(MouseEvent e){}

public void mousePressed(MouseEvent e){}

public void mouseReleased(MouseEvent e){}

public void keyTyped(KeyEvent e)

{ Object source = e.getSource();

actions(source,e.getKeyChar()+"");

}

public void keyPressed(KeyEvent e){}

public void keyReleased(KeyEvent e){}

//===========================================================

//=== IBIO simplified input/output commands - GUI version ===

//===========================================================

public void output(String message)

{ JOptionPane.showMessageDialog(this,message); }

public void outputString(String message)

{ output(message); }

public void output(char info)

{ output(info + ""); }

public void output(byte info)

{ output(info + ""); }

public void output(int info)

{ output(info + ""); }

public void output(long info)

{ output(info + ""); }

public void output(double info)

{ output(info + ""); }

public void output(boolean info)

{ output(info + ""); }

//----- Numerical input methods return 0 on error ------------

public String input(String prompt)

{ return JOptionPane.showInputDialog(this,prompt); }

public String inputString(String prompt)

{ return input(prompt); }

public String input()

{ return input(""); }

public char inputChar(String prompt)

{ char result=(char)0;

try{result=input(prompt).charAt(0);}

catch (Exception e){result = (char)0;}

return result;

}

public byte inputByte(String prompt)

{ byte result=0;

try{result=Byte.valueOf(input(prompt).trim()).byteValue();}

catch (Exception e){result = 0;}

return result;

}

public int inputInt(String prompt)

{ int result=0;

try{result=Integer.valueOf(

input(prompt).trim()).intValue();}

catch (Exception e){result = 0;}

return result;

}

public long inputLong(String prompt)

{ long result=0;

try{result=Long.valueOf(input(prompt).trim()).longValue();}

catch (Exception e){result = 0;}

return result;

}

public double inputDouble(String prompt)

{ double result=0;

try{result=Double.valueOf(

input(prompt).trim()).doubleValue();}

catch (Exception e){result = 0;}

return result;

}

public boolean inputBoolean(String prompt)

{ boolean result=false;

try{result=Boolean.valueOf(

input(prompt).trim()).booleanValue();}

catch (Exception e){result = false;}

return result;

}

//============================================================

//=========== end IBIO =======================================

//============================================================

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class SideChoice extends EasyApp //checked, alright

{

public static void main(String[] args)

{

new SideChoice();}

Label lSideChoice = addLabel("Choose a side",150,30,500,60,this);

Button bT = addButton("T",150,100,100,50,this);

Button bCT = addButton("CT",250,100,100,50,this);

String sideChoice;

public SideChoice() // Constructor - change window appearance

{

setSize(500,250);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

lSideChoice.setFont(new Font("Arial",1,36));

lSideChoice.setBackground(new Color(255,255,180));

lSideChoice.setForeground(Color.blue);

setBackground(new Color(255,255,180));

bT.setFont(new Font("Arial",1,16));

bCT.setFont(new Font("Arial",1,16));

}

public void actions(Object source,String command)

{

if (source == bT) //Allows the player to choose the T side

{

String sideChoice = "TSide"; //String gains the value that will be stored in MySQL

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database [Used Heimetli code with my connection name]

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET SideChoice = ? WHERE idNumCS = 1"; //changes the value of Side in MySQL

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, sideChoice); //changes to the value equal to the string sideChoice

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new MainGameT();

dispose();

}

if (source == bCT) //To choose the CT Side

{

String sideChoice = "CTSide"; //String gets the value of the chosen side to upload it to MySQL

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET SideChoice = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, sideChoice);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new MainGameT(); //Opens a Main Menu when the side is chosen

dispose();} //closes this menu

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class MainGameT extends EasyApp //checked, alrught

{

public static void main(String[] args)

{ new MainGameT();}

Label lSide = addLabel("",60,30,225,60,this);

Button bBuy = addButton("Buy",100,250,266,100,this);

Button bGame = addButton("Game",366,250,266,100,this);

Button bResult = addButton("Result",632,250,266,100,this);

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,400,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,400,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountMGT;

int BalanceMGT;

int RoundCTMGT;

int RoundTMGT;

String MainGameTCondition;

String sqlStringSideChoice;

public MainGameT() // Constructor - change window appearance

{

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountMainGameT = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountMainGameT.next() ){

String sqlRoundCountstringMainGameT = sqlRoundCountMainGameT.getString(1);

RoundCountMGT = Integer.parseInt (sqlRoundCountstringMainGameT);}

ResultSet rsBalanceMainGameT = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalanceMainGameT.next() ){

String rs1BalancestringMainGameT = rsBalanceMainGameT.getString(1);

BalanceMGT = Integer.parseInt (rs1BalancestringMainGameT);}

ResultSet sqlRoundCTMainGameT = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1" ) ;

while( sqlRoundCTMainGameT.next() ){

String sqlRoundCTstringMainGameT = sqlRoundCTMainGameT.getString(1);

RoundCTMGT = Integer.parseInt (sqlRoundCTstringMainGameT);}

ResultSet sqlRoundTMainGameT = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1" ) ;

while( sqlRoundTMainGameT .next() ){

String sqlRoundTstringMainGameT = sqlRoundTMainGameT .getString(1);

RoundTMGT = Integer.parseInt (sqlRoundTstringMainGameT);}

ResultSet sqlMainGameTCondition = stmt.executeQuery( "SELECT MainGameCondition FROM IACompCSGO WHERE idNumCS = 1" ) ;

while( sqlMainGameTCondition .next() ){

MainGameTCondition = sqlMainGameTCondition.getString(1);

}

ResultSet sqlSideChoice = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1" ) ;

while( sqlSideChoice .next() ){

sqlStringSideChoice = sqlSideChoice.getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,500);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

bBuy.setFont(new Font("Arial",1,16));

bGame.setFont(new Font("Arial",1,16));

bResult.setFont(new Font("Arial",1,16));

tRoundNum.setText(RoundCountMGT + "");

tBalance.setText(BalanceMGT + "$");

tCTscore.setText(RoundCTMGT + "");

tTscore.setText(RoundTMGT + "");

if(sqlStringSideChoice.equals("TSide")){

lSide.setText("You are playing for the T side");

} else if(sqlStringSideChoice.equals("CTSide")){

lSide.setText("You are playing for the CT side");

}

if(MainGameTCondition.equals("BuyMenu")){

bBuy.setEnabled(true);

bGame.setEnabled(false);

bResult.setEnabled(false);

}

if(MainGameTCondition.equals("Game")){

bGame.setEnabled(true);

bBuy.setEnabled(false);

bResult.setEnabled(false);

}

if(MainGameTCondition.equals("Result")){

bResult.setEnabled(true);

bBuy.setEnabled(false);

bGame.setEnabled(false);

}

}

public void actions(Object source,String command)

{

if (source == bBuy)

{ new BuyMenu();

bBuy.setEnabled(false);

dispose();}

if (source == bGame){

new GameT();

bGame.setEnabled(false);

dispose();

}

if (source == bResult){

new ResultMenu();

bResult.setEnabled(false);

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class BuyMenu extends EasyApp

{

public static void main(String[] args)

{ new BuyMenu();}

Label lSide = addLabel("",60,30,225,60,this);

Button bPrimaryWeapons = addButton("Weapons",100,100,798,200,this);

Button bWeaponDrop = addButton("Weapon Drop/Pick",100,300,798,100,this);

Button bWeaponChoice = addButton("Weapon Choice",100,400,798,100,this);

Button bQuitBuyMenu = addButton("Go Back",850,600,150,100,this);

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",505,510,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,510,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCTBuyMenu;

int RoundTBuyMenu;

int RoundCountBuyMenu;

int BalanceBuyMenu;

String buyMenuWeapon1PistolsSQLstring;

String sqlStringWeaponChoiceCondition;

String sqlSideChoiceString;

String sqlPistolsConditionString;

String sqlWeaponPickDropString;

public BuyMenu() // Constructor - change window appearance

{

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountBuyMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountBuyMenu.next() ){

String sqlRoundCountstringBuyMenu = sqlRoundCountBuyMenu.getString(1);

RoundCountBuyMenu = Integer.parseInt (sqlRoundCountstringBuyMenu);}

ResultSet rsBalanceBuyMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalanceBuyMenu.next() ){

String rs1BalancestringBuyMenu = rsBalanceBuyMenu.getString(1);

BalanceBuyMenu = Integer.parseInt (rs1BalancestringBuyMenu);}

ResultSet sqlRoundCTBuyMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTBuyMenu.next() ){

String sqlRoundCTstringBuyMenu = sqlRoundCTBuyMenu.getString(1);

RoundCTBuyMenu = Integer.parseInt (sqlRoundCTstringBuyMenu);}

ResultSet sqlRoundTBuyMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTBuyMenu.next() ){

String sqlRoundTstringBuyMenu = sqlRoundTBuyMenu .getString(1);

RoundTBuyMenu = Integer.parseInt (sqlRoundTstringBuyMenu);}

ResultSet sqlWeaponChoiceCondition = stmt.executeQuery( "SELECT WeaponChoiceCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeaponChoiceCondition.next() ){

sqlStringWeaponChoiceCondition = sqlWeaponChoiceCondition.getString(1);}

ResultSet sqlSideChoice = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlSideChoice .next() ){

sqlSideChoiceString = sqlSideChoice.getString(1);

}

//finish

ResultSet sqlPistolCondtion = stmt.executeQuery( "SELECT PistolsCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlPistolCondtion .next() ){

sqlPistolsConditionString = sqlPistolCondtion.getString(1);

}

//finish

ResultSet sqlWeapon1 = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1.next() ){

String buyMenuWeapon1PistolsSQLstring = sqlWeapon1.getString(1);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,700);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountBuyMenu + "");

tBalance.setText(BalanceBuyMenu + "$");

tCTscore.setText(RoundCTBuyMenu + "");

tTscore.setText(RoundTBuyMenu + "");

if(sqlSideChoiceString.equals("CTSide")){

lSide.setText("You are playing for the CT side");

}else if (sqlSideChoiceString.equals("TSide")){

lSide.setText("You are playing for the T side");

}

if(sqlStringWeaponChoiceCondition.equals("False")){

bPrimaryWeapons.setEnabled(false);

bWeaponDrop.setEnabled(false);

bWeaponChoice.setEnabled(true);

}else if (sqlStringWeaponChoiceCondition.equals("True")){

bWeaponChoice.setEnabled(false);

}

if(sqlPistolsConditionString.equals("False")){

bQuitBuyMenu.setEnabled(false);

bWeaponDrop.setEnabled(false);

}else if(sqlPistolsConditionString.equals("True")){

bQuitBuyMenu.setEnabled(true);

bWeaponDrop.setEnabled(true);

}

}

public void actions(Object source,String command)

{

if (source == bWeaponChoice)

{ new Weaponchoice();

dispose();

}

if (source == bWeaponDrop)

{ new buyMenuPickDrop();

dispose();

}

if (source == bPrimaryWeapons)

{

dispose();

new Pistols();

}

if (source == bQuitBuyMenu)

{

new MainGameT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class Weaponchoice extends EasyApp

{

public static void main(String[] args)

{

new Weaponchoice();

}

Label lSide = addLabel("You are playing for T-side",60,30,225,60,this);

Label lSide1 = addLabel("What was your choice before the game for T-side?",60,80,500,30,this);

Label lSide2 = addLabel("What was your choice before the game for CT-side?",600,80,500,30,this);

Label lOR = addLabel("OR",200,190,20,17,this);

Label lOR2 = addLabel("OR",200,290,20,17,this);

Label lOR3 = addLabel("OR",200,390,20,17,this);

Label lOR4 = addLabel("OR",750,190,20,17,this);

Label lOR5 = addLabel("OR",750,290,20,17,this);

Label lOR6 = addLabel("OR",750,390,20,17,this);

Label lOR7 = addLabel("OR",750,490,20,17,this);

Label lOR8 = addLabel("OR",750,590,20,17,this);

Button bChoiceTCZ75 = addButton("CZ75-Auto",50,150,150,100,this);

Button bChoiceTTec9 = addButton("Tec-9",220,150,150,100,this);

Button bChoiceTDesertEagle = addButton("Desert Eagle",50,250,150,100,this);

Button bChoiceTR8Revolver = addButton("R8 Revolver",220,250,150,100,this);

Button bChoiceTMP5SD = addButton("MP5-SD",50,350,150,100,this);

Button bChoiceTMP7 = addButton("MP7",220,350,150,100,this);

Button bChoiceCTUSPS = addButton("USP-S",600,150,150,100,this);

Button bChoiceCTP2000 = addButton("P2000",770,150,150,100,this);

Button bChoiceCTDesertEagle = addButton("Desert Eagle",600,250,150,100,this);

Button bChoiceCTR8Revolver = addButton("R8 Revolver",770,250,150,100,this);

Button bChoiceCTCZ75 = addButton("CZ75-Auto",600,350,150,100,this);

Button bChoiceCTFiveSeveN = addButton("Five-SeveN",770,350,150,100,this);

Button bChoiceCTMP5SD = addButton("MP5-SD",600,450,150,100,this);

Button bChoiceCTMP7 = addButton("MP7",770,450,150,100,this);

Button bChoiceCTM4A4 = addButton("M4A4",600,550,150,100,this);

Button bChoiceCTM4A1S = addButton("M4A1-S",770,550,150,100,this);

Button bQuitWeaponMenu = addButton("Go Back",150,550,150,100,this);

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int WeaponchoiceConditionCount = 0;

static boolean eightweaponcondition = false;

int RoundCTWeaponchoice;

int RoundTWeaponchoice;

int RoundCountWeaponchoice;

int BalanceWeaponchoice;

//new

String TCZ75orTec9string;

String TDeagleORr8string;

String Tmp5SDorMP7string;

String CTuspsORp2000string;

String CTDeagleORr8string;

String CTCZ75or57string;

String CTmp5SDorMP7string;

String CTM4a4orM4a1sstring;

//endnew

public Weaponchoice() // Constructor - change window appearance

{

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountWeaponchoice = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountWeaponchoice.next() ){

String sqlRoundCountstringWeaponchoice = sqlRoundCountWeaponchoice.getString(1);

RoundCountWeaponchoice = Integer.parseInt (sqlRoundCountstringWeaponchoice);}

ResultSet rsBalanceWeaponchoice = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = "+RoundCountWeaponchoice+";" ) ;

// Loop through the result set

while( rsBalanceWeaponchoice.next() ){

String rs1BalancestringWeaponchoice = rsBalanceWeaponchoice.getString(1);

BalanceWeaponchoice = Integer.parseInt (rs1BalancestringWeaponchoice);}

ResultSet sqlRoundCTWeaponchoice = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = "+RoundCountWeaponchoice+";" ) ;

while( sqlRoundCTWeaponchoice.next() ){

String sqlRoundCTstringWeaponchoice = sqlRoundCTWeaponchoice.getString(1);

RoundCTWeaponchoice = Integer.parseInt (sqlRoundCTstringWeaponchoice);}

ResultSet sqlRoundTWeaponchoice = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = "+RoundCountWeaponchoice+";" ) ;

while( sqlRoundTWeaponchoice .next() ){

String sqlRoundTstringWeaponchoice = sqlRoundTWeaponchoice .getString(1);

RoundTWeaponchoice = Integer.parseInt (sqlRoundTstringWeaponchoice);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,700);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountWeaponchoice + "");

tBalance.setText(BalanceWeaponchoice + "$");

tCTscore.setText(RoundCTWeaponchoice + "");

tTscore.setText(RoundTWeaponchoice + "");

bQuitWeaponMenu.setEnabled(false);

}

public void actions(Object source,String command)

{

if (source == bChoiceTCZ75){

bChoiceTTec9.setEnabled(false);

//new1

TCZ75orTec9string = "TCZ75";

//endnew1

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if(source == bChoiceTTec9){

bChoiceTCZ75.setEnabled(false);

//new2

TCZ75orTec9string = "TTec9";

//endnew2

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if (source == bChoiceTDesertEagle){ //Button to choose the Desert Eagle

bChoiceTR8Revolver.setEnabled(false); //The button to choose substitute weapon (R8 Revolver) is disabled

TDeagleORr8string = "TDeagle"; //Sets the value for String to store the user's selection in MySQL later on.

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1; //in order to enable the quit button from this menu, should be equal to 8

}else if(source == bChoiceTR8Revolver){ //Button to choose R8 Revolver

bChoiceTDesertEagle.setEnabled(false); //The button to choose substitute weapon (Desert Eagle ) is disabled

TDeagleORr8string = "TR8"; //Sets the value for String to store the user's selection in MySQL later on.

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1; //in order to enable the quit button from this menu, should be equal to 8

}

if (source == bChoiceTMP5SD){

bChoiceTMP7.setEnabled(false);

//new4

Tmp5SDorMP7string = "TMP5SD";

//endnew4

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}else if(source == bChoiceTMP7){

bChoiceTMP5SD.setEnabled(false);

//new5

Tmp5SDorMP7string = "TMP7";

//endnew5

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if (source == bChoiceCTUSPS){

bChoiceCTP2000.setEnabled(false);

//new6

CTuspsORp2000string = "CTUSPS";

//endnew6

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}else if(source == bChoiceCTP2000){

bChoiceCTUSPS.setEnabled(false);

//new7

CTuspsORp2000string = "CTP2000";

//endnew7

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if (source == bChoiceCTDesertEagle){

bChoiceCTR8Revolver.setEnabled(false);

//new8

CTDeagleORr8string = "CTDeagle";

//endnew8

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}else if(source == bChoiceCTR8Revolver){

bChoiceCTDesertEagle.setEnabled(false);

//new9

CTDeagleORr8string = "CTR8";

//endnew9

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if (source == bChoiceCTCZ75){

bChoiceCTFiveSeveN.setEnabled(false);

//new10

CTCZ75or57string = "CTCZ75";

//endnew10

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}else if(source == bChoiceCTFiveSeveN){

bChoiceCTCZ75.setEnabled(false);

//new11

CTCZ75or57string = "CT57";

//endnew11

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if (source == bChoiceCTMP5SD){

bChoiceCTMP7.setEnabled(false);

//new12

CTmp5SDorMP7string = "CTMP5SD";

//endnew12

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}else if(source == bChoiceCTMP7){

bChoiceCTMP5SD.setEnabled(false);

//new13

CTmp5SDorMP7string = "CTMP7";

//endnew13

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if (source == bChoiceCTM4A4){

bChoiceCTM4A1S.setEnabled(false);

//new14

CTM4a4orM4a1sstring = "CTM4A4";

//endnew14

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}else if(source == bChoiceCTM4A1S){

bChoiceCTM4A4.setEnabled(false);

//new15

CTM4a4orM4a1sstring = "CTM4A1S";

//endnew15

WeaponchoiceConditionCount = WeaponchoiceConditionCount + 1;

}

if(WeaponchoiceConditionCount == 8){

bQuitWeaponMenu.setEnabled(true);

}

if (source == bQuitWeaponMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET WeaponChoiceCondition = ?, TCZ75orTec9 = ?, TDeagleORr8 = ?, Tmp5SDorMP7 = ?, CTuspsORp2000 = ?,CTDeagleORr8 = ?, CTCZ75or57 = ?, CTmp5SDorMP7 = ?, CTM4a4orM4a1s = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

//new 16

preparedStatement.setString(2, TCZ75orTec9string);

preparedStatement.setString(3, TDeagleORr8string);

preparedStatement.setString(4, Tmp5SDorMP7string);

preparedStatement.setString(5, CTuspsORp2000string);

preparedStatement.setString(6, CTDeagleORr8string);

preparedStatement.setString(7, CTCZ75or57string);

preparedStatement.setString(8, CTmp5SDorMP7string);

preparedStatement.setString(9, CTM4a4orM4a1sstring);

//endnew 16

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new BuyMenu();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\* ;

public class Pistols extends EasyApp

{

public static void main(String[] args)

{ new Pistols();

}

Label lSide = addLabel("",60,30,225,60,this);//Adds the label, which states the player's side. Changes later, when the side is imported from MySQL.

Label lGrenadeLabel = addLabel("Grenades",70,90,60,30,this); //Adds the label above the buttons for Grenades

Label lPistolsLabel = addLabel("Pistols",254,90,60,30,this); //Adds the label above the buttons for Pistols

Label lGearLabel = addLabel("Gear",476,90,60,30,this); //Adds the label above the buttons for Gear

Label lSMGLabel = addLabel("SMG",742,90,60,30,this); //Adds the label above the buttons for SMG

Label lRiflesLabel = addLabel("Rifles",1008,90,60,30,this); //Adds the label above the buttons for Rifles

Label lHeavyLabel = addLabel("Heavy",1274,90,60,30,this); //Adds the label above the buttons for Heavy

TextField tWeapon1 = addTextField("",70,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tWeapon2 = addTextField("",170,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tZeus = addTextField("",270,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tGrenade1 = addTextField("",370,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tGrenade2 = addTextField("",470,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tGrenade3 = addTextField("",570,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tGrenade4 = addTextField("",670,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tKits = addTextField("",770,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

TextField tArmor = addTextField("",870,820,100,30,this); //Adds the text field, where player's current weapon is displayed. Imported from MySQL

Button bMolotov = addButton("Molotov\n400$",0,120,183,100, this); //Adds the button, which allows to buy Molotov

Button bDecoy = addButton("Decoy\n50$",0,220,183,100, this); //Adds the button, which allows to buy Decoy

Button bFlashbang = addButton("Flashbang\n200$",0,320,183,100,this); //Adds the button, which allows to buy Flashbang

Button bHEGrenade = addButton("HE Grenade\n300$",0,420,183,100,this); //Adds the button, which allows to buy HE Grendade

Button bSmoke = addButton("Smoke\n600$",0,520,183,100,this); //Adds the button, which allows to buy Smoke

Button bIncendiary = addButton("Incendiary\n600$",0,120,183,100, this); //Adds the button, which allows to buy Incendiary

Button bGlock18 = addButton("Glock-18\n200$",183,120,183,100,this); //Adds the button, which allows to buy Glock-18

Button bDualBerettas= addButton("Dual Berettas\n300$",183,220,183,100,this); //Adds the button, which allows to buy Dual Berettas

Button bP250 = addButton("P250\n300$",183,320,183,100,this); //Adds the button, which allows to buy P250

Button bCZ75Auto = addButton("CZ75 - Auto\n500$",183,420,183,100,this); //Adds the button, which allows to buy CZ75 - Auto

Button bDesertEagle = addButton("Desert Eagle\n700$",183,520,183,100,this); //Adds the button, which allows to buy Desert Eagle

Button bR8Revolver = addButton("R8 Revolver\n600$",183,620,183,100,this); //Adds the button, which allows to buy R8 Revolver

Button bTec9 = addButton("Tec-9\n500$",183,720,183,100,this); //Adds the button, which allows to buy Tec-9

Button bUSPS = addButton("USP-S\n200$",183,120,183,100,this); //Adds the button, which allows to buy USP-S

Button bP2000 = addButton("P2000\n200$",183,120,183,100,this); //Adds the button, which allows to buy P2000

Button bFiveSeven = addButton("Five-SeveN\n500$",183,420,183,100,this); //CT WEAPON CHOICE TO DO

Button bKevlarVest = addButton("Kevlar Vest\n650$",366,120,266,100,this);

Button bKevlarVestHelmet = addButton("Kevlar Vest + Helmet\n1000$",366,220,266,100,this);

Button bZeusx27 = addButton("Zeus x27\n200$",366,320,266,100,this);

Button bHelmet = addButton("Helmet (when armor is full)\n350$",366,420,266,100,this);

Button bKits = addButton("Defuse Kits\n400$",366,520,266,100,this); //finish

Button bMAC10 = addButton("MAC-10\n1050$",632,120,266,100,this);

Button bMP5SD = addButton("MP5-SD\n1500$",632,220,266,100,this);

Button bMP7 = addButton("MP7\n1500$",632,320,266,100,this);

Button bUMP45 = addButton("UMP - 45\n1200$",632,420,266,100,this);

Button bP90 = addButton("P90\n2350$",632,520,266,100,this);

Button bPPBizon = addButton("PP-Bizon\n1400$",632,620,266,100,this);

Button bMP9 = addButton("MP9\n1250$",632,120,266,100,this); //CT TO DO

Button bGalilAR = addButton("Galil AR\n1800$",898,120,266,100,this);

Button bAK47 = addButton("AK-47\n2700$",898,220,266,100,this);

Button bSSG08 = addButton("SSG 08\n1700$",898,320,266,100,this);

Button bSG553 = addButton("SG 553\n3000$",898,420,266,100,this);

Button bAWP = addButton("AWP\n4750$",898,520,266,100,this);

Button bG3SG1 = addButton("G3SG1\n5000$",898,620,266,100,this);

Button bM4A4 = addButton("M4A4\n3100$",898,220,266,100,this); //CT WEAPON CHOICE TO DO

Button bM4A1S = addButton("M4A1-S\n2900$",898,220,266,100,this); //CT WEAPON CHOICE TO DO

Button bAUG = addButton("AUG\n3300$",898,420,266,100,this);

Button bFAMAS = addButton("FAMAS\n2050$",898,120,266,100,this); //CT WEAPON CHOICE TO DO

Button bSCAR20 = addButton("SCAR-20\n5000$",898,620,266,100,this); //CT WEAPON CHOICE TO DO

Button bNova = addButton("Nova\n1050$",1164,120,266,100,this);

Button bXM1014 = addButton("XM1014\n2000$",1164,220,266,100,this);

Button bSawedOff = addButton("Sawed-Off\n1100$",1164,320,266,100,this);

Button bM249 = addButton("M249\n5200$",1164,420,266,100,this);

Button bNegev= addButton("Negev\n1700$",1164,520,266,100,this);

Button bQuitPistolsMenu = addButton("Go Back",1250,700,150,100,this);

TextField tCTscore = addTextField("0",350,50,31,30,this); //Shows how many rounds has the CT side won

TextField tTscore = addTextField("0",382,50,31,30,this); //Shows how many rounds has the T side won

TextField tRoundNum = addTextField("0",960,900,31,30,this); //Shows how many rounds were played

TextField tBalance = addTextField("0",900,50,62,30,this); //Shows the balance of the player

Label lCT = addLabel("CT",330,50,50,30,this); //Defines that the text field is showing how many rounds has the CT side won

Label lT = addLabel("T",420,50,50,30,this); //Defines that the text field is showing how many rounds has the T side won

Label lRoundNum = addLabel("Round: ",900,900,50,30,this); //Defines that the text field is showing how many rounds were played

Label lBalance = addLabel("Balance: ",848,50,62,30,this); //Defines that the text field is showing the balance of the player

int grenadeCountPistols = 0;

int flashbangCountPistols = 0;

boolean playerHasFullArmor = false;

boolean playerHasFullArmorAndHelmet = false;

int intRSBalance;

int intSQLRoundCT;

int intSQLRoundT;

int intSQLRoundNumber;

String Weapon1PistolsSQL;

String Weapon2PistolsSQL;

String ArmorPistolsSQL;

String ZeusPistolsSQL;

String Grenade1PistolsSQL;

String Grenade2PistolsSQL;

String Grenade3PistolsSQL;

String Grenade4PistolsSQL;

String kitsStringPistolsSQL;

String sqlStringTCZ75orTec9;

String sqlStringTDeagleORr8;

String sqlStringTmp5SDorMP7;

String sqlStringCTuspsORp2000;

String sqlStringCTDeagleORr8;

String sqlStringsqlCTCZ75or57;

String sqlStringCTmp5SDorMP7;

String sqlStringCTM4a4orM4a1s;

String sqlStringMolotovBuyCondition;

String sqlStringDecoyBuyCondition;

String sqlStringFlashBuyCondition;

String sqlStringSmokeBuyCondition;

String sqlStringHEBuyCondition;

String sqlStringIncendiaryBuyCondition;

String sqlStringLimitGrenadeCondition;

String sqlStringSide;

String sqlStringPistolNullCondition;

//mySQL

public Pistols() // Constructor - change window appearance

{

//MySQL

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCount = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCount.next() ){

String sqlRoundCountstring = sqlRoundCount.getString(1);

intSQLRoundNumber = Integer.parseInt (sqlRoundCountstring);}

ResultSet rsBalance = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalance.next() ){

String rs1Balancestring = rsBalance.getString(1);

intRSBalance = Integer.parseInt (rs1Balancestring);}

ResultSet sqlRoundCT = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCT.next() ){

String sqlRoundCTstring = sqlRoundCT.getString(1);

intSQLRoundCT = Integer.parseInt (sqlRoundCTstring);}

ResultSet sqlRoundT = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundT.next() ){

String sqlRoundTstring = sqlRoundT.getString(1);

intSQLRoundT = Integer.parseInt (sqlRoundTstring);}

ResultSet sqlWeapon1 = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1.next() ){

Weapon1PistolsSQL = sqlWeapon1.getString(1);}

ResultSet sqlWeapon2 = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2.next() ){

Weapon2PistolsSQL = sqlWeapon2.getString(1);}

ResultSet sqlArmor = stmt.executeQuery( "SELECT Armor FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlArmor.next() ){

ArmorPistolsSQL = sqlArmor.getString(1);}

ResultSet sqlZeus = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;") ;

while( sqlZeus.next() ){

ZeusPistolsSQL = sqlZeus.getString(1);}

ResultSet kitsPistolsSQL = stmt.executeQuery( "SELECT kits FROM IACompCSGO WHERE idNumCS = 1;") ;

while( kitsPistolsSQL.next() ){

kitsStringPistolsSQL = kitsPistolsSQL.getString(1);}

ResultSet sqlGrenade1 = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1.next() ){

Grenade1PistolsSQL = sqlGrenade1.getString(1);}

ResultSet sqlGrenade2 = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2.next() ){

Grenade2PistolsSQL = sqlGrenade2.getString(1);}

ResultSet sqlGrenade3 = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;") ;

while( sqlGrenade3.next() ){

Grenade3PistolsSQL = sqlGrenade3.getString(1);}

ResultSet sqlGrenade4 = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4.next() ){

Grenade4PistolsSQL = sqlGrenade4.getString(1);}

ResultSet sqlTCZ75orTec9 = stmt.executeQuery( "SELECT TCZ75orTec9 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlTCZ75orTec9.next() ){

sqlStringTCZ75orTec9 = sqlTCZ75orTec9.getString(1);}

ResultSet sqlTDeagleORr8 = stmt.executeQuery( "SELECT TDeagleORr8 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlTDeagleORr8.next() ){

sqlStringTDeagleORr8 = sqlTDeagleORr8.getString(1);}

ResultSet sqlTmp5SDorMP7 = stmt.executeQuery( "SELECT Tmp5SDorMP7 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlTmp5SDorMP7.next() ){

sqlStringTmp5SDorMP7 = sqlTmp5SDorMP7.getString(1);}

ResultSet sqlCTuspsORp2000 = stmt.executeQuery( "SELECT CTuspsORp2000 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlCTuspsORp2000.next() ){

sqlStringCTuspsORp2000 = sqlCTuspsORp2000.getString(1);}

ResultSet sqlCTDeagleORr8 = stmt.executeQuery( "SELECT CTDeagleORr8 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlCTDeagleORr8.next() ){

sqlStringCTDeagleORr8 = sqlCTDeagleORr8.getString(1);}

ResultSet sqlCTCZ75or57 = stmt.executeQuery( "SELECT CTCZ75or57 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlCTCZ75or57.next() ){

sqlStringsqlCTCZ75or57 = sqlCTCZ75or57.getString(1);}

ResultSet sqlCTmp5SDorMP7 = stmt.executeQuery( "SELECT CTmp5SDorMP7 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlCTmp5SDorMP7.next() ){

sqlStringCTmp5SDorMP7 = sqlCTmp5SDorMP7.getString(1);}

ResultSet sqlCTM4a4orM4a1s = stmt.executeQuery( "SELECT CTM4a4orM4a1s FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlCTM4a4orM4a1s.next() ){

sqlStringCTM4a4orM4a1s = sqlCTM4a4orM4a1s.getString(1);}

ResultSet sqlMolotovBuyCondition = stmt.executeQuery( "SELECT MolotovBuyCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlMolotovBuyCondition.next() ){

sqlStringMolotovBuyCondition = sqlMolotovBuyCondition.getString(1);}

ResultSet sqlDecoyBuyCondition = stmt.executeQuery( "SELECT DecoyBuyCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlDecoyBuyCondition.next() ){

sqlStringDecoyBuyCondition = sqlDecoyBuyCondition.getString(1);}

ResultSet sqlFlashBuyCondition = stmt.executeQuery( "SELECT FlashBuyCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlFlashBuyCondition.next() ){

sqlStringFlashBuyCondition = sqlFlashBuyCondition.getString(1);}

ResultSet sqlSmokeBuyCondition = stmt.executeQuery( "SELECT SmokeBuyCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlSmokeBuyCondition.next() ){

sqlStringSmokeBuyCondition = sqlSmokeBuyCondition.getString(1);}

ResultSet sqlHEBuyCondition = stmt.executeQuery( "SELECT HEBuyCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlHEBuyCondition.next() ){

sqlStringHEBuyCondition = sqlHEBuyCondition.getString(1);}

ResultSet sqlIncendiaryBuyCondition = stmt.executeQuery( "SELECT incendiaryBuyCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlIncendiaryBuyCondition.next() ){

sqlStringIncendiaryBuyCondition = sqlIncendiaryBuyCondition.getString(1);}

ResultSet sqlLimitGrenadeCondition = stmt.executeQuery( "SELECT LimitGrenadeCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlLimitGrenadeCondition.next() ){

sqlStringLimitGrenadeCondition = sqlLimitGrenadeCondition.getString(1);}

ResultSet sqlPistolNullCondition = stmt.executeQuery( "SELECT PistolNullCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlPistolNullCondition.next() ){

sqlStringPistolNullCondition = sqlPistolNullCondition.getString(1);}

ResultSet sqlSide = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlSide.next() ){

sqlStringSide = sqlSide.getString(1);}

//finish it

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

if((intSQLRoundNumber == 1 && Weapon1PistolsSQL == null && sqlStringPistolNullCondition.equals("False") && sqlStringSide.equals("TSide") ) || (intSQLRoundNumber == 16 && Weapon1PistolsSQL == null && sqlStringPistolNullCondition.equals("False") && sqlStringSide.equals("TSide")) ){

Weapon1PistolsSQL = "Glock-18";

bGlock18.setEnabled(false);

}

if((intSQLRoundNumber == 1 && Weapon1PistolsSQL == null && sqlStringPistolNullCondition.equals("False") && sqlStringSide.equals("CTSide") && sqlStringCTuspsORp2000.equals("CTUSPS")) || (intSQLRoundNumber == 16 && Weapon1PistolsSQL == null && sqlStringPistolNullCondition.equals("False") && sqlStringSide.equals("CTSide") && sqlStringCTuspsORp2000.equals("CTUSPS")) ){

Weapon1PistolsSQL = "USP-S";

bUSPS.setEnabled(false);

}

if((intSQLRoundNumber == 1 && Weapon1PistolsSQL == null && sqlStringPistolNullCondition.equals("False") && sqlStringSide.equals("CTSide") && sqlStringCTuspsORp2000.equals("P2000")) || (intSQLRoundNumber == 16 && Weapon1PistolsSQL == null && sqlStringPistolNullCondition.equals("False") && sqlStringSide.equals("CTSide") && sqlStringCTuspsORp2000.equals("P2000")) ){

Weapon1PistolsSQL = "P2000";

bP2000.setEnabled(false);

}

tWeapon1.setText(Weapon1PistolsSQL + "");

tWeapon2.setText(Weapon2PistolsSQL + "");

tZeus.setText(ZeusPistolsSQL + "");

tGrenade1.setText(Grenade1PistolsSQL + "");

tGrenade2.setText(Grenade2PistolsSQL + "");

tGrenade3.setText(Grenade3PistolsSQL + "");

tGrenade4.setText(Grenade4PistolsSQL + "");

tKits.setText(kitsStringPistolsSQL + "");

tArmor.setText(ArmorPistolsSQL + "");

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(intSQLRoundNumber + ""); //shows the number of rounds played

tBalance.setText(intRSBalance + "$"); //shows the balance of the player

tCTscore.setText(intSQLRoundCT + ""); //shows the number of rounds that the CT side have won

tTscore.setText(intSQLRoundT + ""); //shows the number of rounds that the T side have won

if(sqlStringSide.equals("TSide")){

lSide.setText("You are playing for T side");

}

if(sqlStringSide.equals("CTSide")){

lSide.setText("You are playing for CT side");

}

if(Grenade1PistolsSQL != null && Grenade2PistolsSQL != null && Grenade3PistolsSQL != null && Grenade4PistolsSQL != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

bHelmet.setEnabled(false);

if(sqlStringSide.equals("TSide")){ //To remove the weapons, that are not available and taking the same slot, when the player is playing for T Side

bGlock18.setVisible(true); //To make the purchase of Glock-18 available, as it is weapon exclusive for T Side

bMolotov.setVisible(true); //To make the purchase of Molotov available, as it is weapon exclusive for T Side

bTec9.setVisible(true); //To make the purchase of Tec-9 available, as it is weapon exclusive for T Side

bMAC10.setVisible(true); //To make the purchase of MAC-10 available, as it is weapon exclusive for T Side

bGalilAR.setVisible(true); //To make the purchase of Galil AR available, as it is weapon exclusive for T Side

bAK47.setVisible(true); //To make the purchase of AK-47 available, as it is weapon exclusive for T Side

bG3SG1.setVisible(true); //To make the purchase of G3SG1 available, as it is weapon exclusive for T Side

bSG553.setVisible(true); //To make the purchase of SG553 available, as it is weapon exclusive for T Side

bKits.setVisible(false); //To make the purchase of Kits not available, as it is equipment exclusive for CT Side

bMP9.setVisible(false); //To make the purchase of MP9 not available, as it is equipment exclusive for CT Side

bFiveSeven.setVisible(false); //To make the purchase of Five-SeveN not available, as it is equipment exclusive for CT Side

bFAMAS.setVisible(false); //To make the purchase of FAMAS not available, as it is weapon exclusive for CT Side

bSCAR20.setVisible(false); //To make the purchase of SCAR-20 not available, as it is weapon exclusive for CT Side

bAUG.setVisible(false); //To make the purchase of AUG not available, as it is weapon exclusive for CT Side

bIncendiary.setVisible(false); //To make the purchase of Incendiary not available, as it is weapon exclusive for CT Side

if(sqlStringTCZ75orTec9.equals("TCZ75")){ //If statement for weapons chosen in Weaponchoice menu. One - to disable, another - enable

bTec9.setEnabled(false);

}else if(sqlStringTCZ75orTec9.equals("TTec9")){

bCZ75Auto.setEnabled(false);

}

if(sqlStringTDeagleORr8.equals("TDeagle")){ //If statement for weapons chosen in Weaponchoice menu. One - to disable, another - enable

bR8Revolver.setEnabled(false);

}else if(sqlStringTDeagleORr8.equals("TR8")){

bDesertEagle.setEnabled(false);

}

if(sqlStringTmp5SDorMP7.equals("TMP5SD")){ //If statement for weapons chosen in Weaponchoice menu. One - to disable, another - enable

bMP7.setEnabled(false);

}else if(sqlStringTmp5SDorMP7.equals("TMP7")){

bMP5SD.setEnabled(false);

}

}

if(sqlStringSide.equals("CTSide")){ //To remove the weapons, that are not available and taking the same slot, when the player is playing for T Side

bGlock18.setVisible(false); //To make the purchase of Glock-18 not available, as it is weapon exclusive for T Side

bMolotov.setVisible(false); //To make the purchase of Molotov not available, as it is weapon exclusive for T Side

bTec9.setVisible(false); //To make the purchase of Tec-9 not available, as it is weapon exclusive for T Side

bMAC10.setVisible(false);

bGalilAR.setVisible(false);

bAK47.setVisible(false);

bG3SG1.setVisible(false);

bKits.setVisible(true);

bMP9.setVisible(true);

bFiveSeven.setVisible(true);

bFAMAS.setVisible(true);

bSCAR20.setVisible(true);

bAUG.setVisible(true);

bSG553.setVisible(false);

bIncendiary.setVisible(true);

if(sqlStringCTuspsORp2000.equals("CTUSPS")){

bUSPS.setVisible(true);

bP2000.setVisible(false);

} else if(sqlStringCTuspsORp2000.equals("CTP2000")){

bP2000.setVisible(true);

bUSPS.setVisible(false);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle")){

bR8Revolver.setEnabled(false);

bDesertEagle.setEnabled(true);

}else if(sqlStringCTDeagleORr8.equals("CTR8")){

bDesertEagle.setEnabled(false);

bR8Revolver.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75")){

bCZ75Auto.setVisible(true);

bFiveSeven.setVisible(false);

}else if(sqlStringsqlCTCZ75or57.equals("CT57")){

bFiveSeven.setVisible(true);

bCZ75Auto.setVisible(false);

}

if(sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP7.setEnabled(false);

bMP5SD.setEnabled(true);

}else if(sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

bMP5SD.setEnabled(false);

}

if(sqlStringCTM4a4orM4a1s.equals("CTM4A4")){

bM4A4.setVisible(true);

bM4A1S.setVisible(false);

}else if(sqlStringCTM4a4orM4a1s.equals("CTM4A1S")){

bM4A4.setVisible(false);

bM4A1S.setVisible(true);

}

}

if(intRSBalance < 5200){

bM249.setEnabled(false);

}

if(intRSBalance < 5000){

bG3SG1.setEnabled(false);

bSCAR20.setEnabled(false);

}

if(intRSBalance < 4750){

bAWP.setEnabled(false);

}

if(intRSBalance < 3300){

bAUG.setEnabled(false);

}

if(intRSBalance < 3100){

bM4A4.setEnabled(false);

}

if(intRSBalance < 3000){

bSG553.setEnabled(false);

}

if(intRSBalance < 2900){

bM4A1S.setEnabled(false);

}

if(intRSBalance < 2700){

bAK47.setEnabled(false);

}

if(intRSBalance < 2350){

bP90.setEnabled(false);

}

if(intRSBalance < 2050){

bFAMAS.setEnabled(false);

}

if(intRSBalance < 2000){

bXM1014.setEnabled(false);

}

if(intRSBalance < 1800){

bGalilAR.setEnabled(false);

}

if(intRSBalance < 1700){

bSSG08.setEnabled(false);

bNegev.setEnabled(false);

}

if(intRSBalance < 1500){

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

}

if(intRSBalance < 1400){

bPPBizon.setEnabled(false);

}

if(intRSBalance < 1250){

bMP9.setEnabled(false);

}

if(intRSBalance < 1200){

bUMP45.setEnabled(false);

}

if(intRSBalance < 1100){

bSawedOff.setEnabled(false);

}

if(intRSBalance < 1050){

bMAC10.setEnabled(false);

bNova.setEnabled(false);

}

if(intRSBalance < 1000){

bKevlarVestHelmet.setEnabled(false);

}

if(intRSBalance < 700){

bDesertEagle.setEnabled(false);

}

if(intRSBalance < 650){

bKevlarVest.setEnabled(false);

}

if(intRSBalance < 600){

bR8Revolver.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(intRSBalance < 500){

bCZ75Auto.setEnabled(false);

bTec9.setEnabled(false);

bFiveSeven.setEnabled(false);

}

if(intRSBalance < 400){

bMolotov.setEnabled(false);

bKits.setEnabled(false);

}

if(intRSBalance < 350){

bHelmet.setEnabled(false);

}

if(intRSBalance < 300){

bDualBerettas.setEnabled(false);

bP250.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

}

if(intRSBalance < 200){

bGlock18.setEnabled(false);

bZeusx27.setEnabled(false);

bFlashbang.setEnabled(false);

bP2000.setEnabled(false);

bUSPS.setEnabled(false);

}

if(intRSBalance < 50){

bDecoy.setEnabled(false);

}

if(sqlStringMolotovBuyCondition.equals("True")){

bMolotov.setEnabled(false);

}

if(sqlStringDecoyBuyCondition.equals("True")){

bDecoy.setEnabled(false);

}

if(sqlStringFlashBuyCondition.equals("True")){

bFlashbang.setEnabled(false);

}

if(sqlStringSmokeBuyCondition.equals("True")){

bSmoke.setEnabled(false);

}

if(sqlStringHEBuyCondition.equals("True")){

bHEGrenade.setEnabled(false);

}

if(sqlStringIncendiaryBuyCondition.equals("True")){

bIncendiary.setEnabled(false);

}

if(sqlStringLimitGrenadeCondition.equals("True")){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

bHEGrenade.setEnabled(false);

}

if(ArmorPistolsSQL != null){ //finish later

if(ArmorPistolsSQL.equals("Kevlar Vest")){

bKevlarVest.setEnabled(false);

bKevlarVestHelmet.setEnabled(false);

if(intRSBalance > 350){

bHelmet.setEnabled(true);

}else{

bHelmet.setEnabled(false);

}

}

if(ArmorPistolsSQL.equals("Kevlar Vest + Helmet")){

bKevlarVest.setEnabled(false);

bKevlarVestHelmet.setEnabled(false);

bHelmet.setEnabled(false);

}

if(ArmorPistolsSQL.equals(null)){

bHelmet.setEnabled(false);

}

}

if(ZeusPistolsSQL != null){ //ALLIAS02 DONE FOR KITS

if(ZeusPistolsSQL.equals("Zeus")){

bZeusx27.setEnabled(false);

}

}

if(kitsStringPistolsSQL != null){ //ALLIAS02 DONE FOR KITS

if(kitsStringPistolsSQL.equals("Kits")){

bKits.setEnabled(false);

}

}

if(Weapon1PistolsSQL != null){ //doNE for CT

if(Weapon1PistolsSQL.equals("Glock-18")){

bGlock18.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("Dual Berettas")){

bDualBerettas.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("P250")){

bP250.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("CZ75-Auto")){

bCZ75Auto.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("Desert Eagle")){

bDesertEagle.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("R8 Revolver")){

bR8Revolver.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("Tec-9")){

bTec9.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("Five-SeveN")){

bFiveSeven.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("USP-S")){

bUSPS.setEnabled(false);

}

if(Weapon1PistolsSQL.equals("P2000")){

bP2000.setEnabled(false);

}

}

if(Weapon2PistolsSQL != null){

if(Weapon2PistolsSQL.equals("MAC-10")){

bMAC10.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("MP5-SD")){

bMP5SD.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("MP7")){

bMP7.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("UMP-45")){

bUMP45.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("P90")){

bP90.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("PP-Bizon")){

bPPBizon.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("Galil AR")){

bGalilAR.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("AK-47")){

bAK47.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("SSG 08")){

bSSG08.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("SG 553")){

bSG553.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("AWP")){

bAWP.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("G3SG1")){

bG3SG1.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("Nova")){

bNova.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("XM1014")){

bXM1014.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("Sawed-Off")){

bSawedOff.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("M249")){

bM249.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("Negev")){

bNegev.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("MP9")){

bMP9.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("M4A4")){

bM4A4.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("M4A1-S")){

bM4A1S.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("AUG")){

bAUG.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("FAMAS")){

bFAMAS.setEnabled(false);

}

if(Weapon2PistolsSQL.equals("SCAR-20")){

bSCAR20.setEnabled(false);

}

}

}

public void actions(Object source,String command){

//DONE ???

if(source == bMAC10){ //done

intRSBalance = intRSBalance - 1050;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "MAC-10";

bMAC10.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bMP5SD){ //The user decides to buy MP5-SD

intRSBalance = intRSBalance - 1500; //Makes a calculation: (Balance - the price of the weapon (here: MP5-SD = 1500)) = remaining balance

tBalance.setText(intRSBalance + "$"); //updates the balance textfield in order to show the player's current balance after the purchase

Weapon2PistolsSQL = "MP5-SD"; //sets the player's weapon equal to MP5-SD. When quitting the menu, updating it in MySQL

bMP5SD.setEnabled(false); //disables the button of the weapon (MP5-SD), as it was already purchased

if(intRSBalance >= 5200){ //allows the purchase of the weapon, if there is enough balance left

bM249.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){ //allows the purchase of the weapon, if there is enough balance left. Only for T-Side exclusive weapons

if(intRSBalance >= 5000){ //allows the purchase of the weapon, if there is enough balance left

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){ //allows the purchase of the weapon, if there is enough balance left

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){ //allows the purchase of the weapon, if there is enough balance left

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){ //allows the purchase of the weapon, if there is enough balance left

bMAC10.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){ //allows the purchase of the weapon, if there is enough balance left. Only for CT-Side exclusive weapons

if(intRSBalance >= 5000){ //allows the purchase of the weapon, if there is enough balance left

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){ //allows the purchase of the weapon, if there is enough balance left

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){ //allows the purchase of the weapon, if there is enough balance left

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){ //allows the purchase of the weapon, if there is enough balance left

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){ //allows the purchase of the weapon, if there is enough balance left

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){ //allows the purchase of the weapon, if there is enough balance left

bMP9.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bMP7){ //done

intRSBalance = intRSBalance - 1500;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "MP7";

bMP7.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bUMP45){ //done

intRSBalance = intRSBalance - 1200;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "UMP-45";

bUMP45.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bP90){ //done

intRSBalance = intRSBalance - 2350;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "P90";

bP90.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

}

if(source == bPPBizon){ //done

intRSBalance = intRSBalance - 1400;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "PP-Bizon";

bPPBizon.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

}

//Rifles

if(source == bGalilAR){ //done

intRSBalance = intRSBalance - 1800;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "Galil AR";

bGalilAR.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

}

if(source == bAK47){ //done

intRSBalance = intRSBalance - 2700;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "AK-47";

bAK47.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

bNova.setEnabled(true);

}

}

if(source == bSSG08){ //done

intRSBalance = intRSBalance - 1700;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "SSG 08";

bSSG08.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

}

if(source == bSG553){ //done

intRSBalance = intRSBalance - 3000;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "SG 553";

bSG553.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

bNova.setEnabled(true);

}

}

if(source == bAWP){

intRSBalance = intRSBalance - 4750;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "AWP";

bAWP.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

}

if(source == bG3SG1){ //done

intRSBalance = intRSBalance - 5000;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "G3SG1";

bG3SG1.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

bNova.setEnabled(true);

}

}

//Heavy

if(source == bNova){ //done

intRSBalance = intRSBalance - 1050;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "Nova";

bNova.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

}

if(source == bXM1014){ //done

intRSBalance = intRSBalance - 2000;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "XM1014";

bXM1014.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bSawedOff){ //done

intRSBalance = intRSBalance - 1100;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "Sawed-Off";

bXM1014.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bM249){ //done

intRSBalance = intRSBalance - 5200;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "M249";

bM249.setEnabled(false);

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

if(source == bNegev){ //done

intRSBalance = intRSBalance - 1700;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "Negev";

bNegev.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 5000){

bG3SG1.setEnabled(true);

}

if(intRSBalance >= 2700){

bAK47.setEnabled(true);

}

if(intRSBalance >= 1800){

bGalilAR.setEnabled(true);

}

if(intRSBalance >= 1050){

bMAC10.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringTmp5SDorMP7.equals("TMP7")){

bMP7.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 3000){

bSG553.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

}

//WEAPONS FOR CT

if(source == bMP9){ //DONE

intRSBalance = intRSBalance - 1250;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "MP9";

bMP9.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

}

if(source == bAUG){ //DONE

intRSBalance = intRSBalance - 3300;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "AUG";

bAUG.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

}

if(source == bM4A4){ //DONE

intRSBalance = intRSBalance - 3100;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "M4A4";

bM4A4.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

}

if(source == bM4A1S){ //DONE

intRSBalance = intRSBalance - 2900;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "M4A1-S";

bM4A1S.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

}

if(source == bFAMAS){ //DONE

intRSBalance = intRSBalance - 2050;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "FAMAS";

bFAMAS.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 5000){

bSCAR20.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

}

if(source == bSCAR20){ //DONE

intRSBalance = intRSBalance - 5000;

tBalance.setText(intRSBalance + "$");

Weapon2PistolsSQL = "SCAR-20";

bSCAR20.setEnabled(false);

if(intRSBalance >= 5200){

bM249.setEnabled(true);

}

if(intRSBalance >= 4750){

bAWP.setEnabled(true);

}

if(intRSBalance >= 2350){

bP90.setEnabled(true);

}

if(intRSBalance >= 2000){

bXM1014.setEnabled(true);

}

if(intRSBalance >= 1700){

bSSG08.setEnabled(true);

bNegev.setEnabled(true);

}

if(intRSBalance >= 1400){

bPPBizon.setEnabled(true);

}

if(intRSBalance >= 1200){

bUMP45.setEnabled(true);

}

if(intRSBalance >= 1100){

bSawedOff.setEnabled(true);

}

if(intRSBalance >= 1050){

bNova.setEnabled(true);

}

if(intRSBalance >= 3300){

bAUG.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP5SD")){

bMP5SD.setEnabled(true);

}

if(intRSBalance >= 1500 && sqlStringCTmp5SDorMP7.equals("CTMP7")){

bMP7.setEnabled(true);

}

if(intRSBalance >= 1250){

bMP9.setEnabled(true);

}

if(intRSBalance >= 3100){

bM4A4.setEnabled(true);

}

if(intRSBalance >= 2050){

bFAMAS.setEnabled(true);

}

if(intRSBalance >= 2900){

bM4A1S.setEnabled(true);

}

}

//Grenades

if(source == bMolotov){

intRSBalance = intRSBalance - 400;

tBalance.setText(intRSBalance + "$");

if(Grenade1PistolsSQL == null){

Grenade1PistolsSQL = "Molotov";

}else if(Grenade2PistolsSQL == null){

Grenade2PistolsSQL = "Molotov";

}else if(Grenade3PistolsSQL == null){

Grenade3PistolsSQL = "Molotov";

}else if(Grenade4PistolsSQL == null){

Grenade4PistolsSQL = "Molotov";

}

bMolotov.setEnabled(false);

grenadeCountPistols = grenadeCountPistols + 1;

sqlStringMolotovBuyCondition = "True";

if(grenadeCountPistols == 4){

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

sqlStringLimitGrenadeCondition = "True";

}

}

if(source == bIncendiary){

intRSBalance = intRSBalance - 600;

tBalance.setText(intRSBalance + "$");

if(Grenade1PistolsSQL == null){

Grenade1PistolsSQL = "Incendiary";

}else if(Grenade2PistolsSQL == null){

Grenade2PistolsSQL = "Incendiary";

}else if(Grenade3PistolsSQL == null){

Grenade3PistolsSQL = "Incendiary";

}else if(Grenade4PistolsSQL == null){

Grenade4PistolsSQL = "Incendiary";

}

bIncendiary.setEnabled(false);

grenadeCountPistols = grenadeCountPistols + 1;

sqlStringIncendiaryBuyCondition = "True";

if(grenadeCountPistols == 4){

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

sqlStringLimitGrenadeCondition = "True";

}

}

if(source == bDecoy){

intRSBalance = intRSBalance - 50;

tBalance.setText(intRSBalance + "$");

if(Grenade1PistolsSQL == null){

Grenade1PistolsSQL = "Decoy";

}else if(Grenade2PistolsSQL == null){

Grenade2PistolsSQL = "Decoy";

}else if(Grenade3PistolsSQL == null){

Grenade3PistolsSQL = "Decoy";

}else if(Grenade4PistolsSQL == null){

Grenade4PistolsSQL = "Decoy";

}

bDecoy.setEnabled(false);

grenadeCountPistols = grenadeCountPistols + 1;

sqlStringDecoyBuyCondition = "True";

if(grenadeCountPistols == 4){

bMolotov.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

sqlStringLimitGrenadeCondition = "True";

}

}

if(source == bFlashbang){

intRSBalance = intRSBalance - 200;

tBalance.setText(intRSBalance + "$");

if(Grenade1PistolsSQL == null){

Grenade1PistolsSQL = "Flashbang";

}else if(Grenade2PistolsSQL == null){

Grenade2PistolsSQL = "Flashbang";

}else if(Grenade3PistolsSQL == null){

Grenade3PistolsSQL = "Flashbang";

}else if(Grenade4PistolsSQL == null){

Grenade4PistolsSQL = "Flashbang";

}

grenadeCountPistols = grenadeCountPistols + 1;

flashbangCountPistols = flashbangCountPistols + 1;

if(flashbangCountPistols == 2){

bFlashbang.setEnabled(false);

sqlStringFlashBuyCondition = "True";

}

if(grenadeCountPistols == 4){

bMolotov.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bDecoy.setEnabled(false);

bIncendiary.setEnabled(false);

sqlStringLimitGrenadeCondition = "True";

}

}

if(source == bHEGrenade){

intRSBalance = intRSBalance - 300;

tBalance.setText(intRSBalance + "$");

if(Grenade1PistolsSQL == null){

Grenade1PistolsSQL = "HE Grenade";

}else if(Grenade2PistolsSQL == null){

Grenade2PistolsSQL = "HE Grenade";

}else if(Grenade3PistolsSQL == null){

Grenade3PistolsSQL = "HE Grenade";

}else if(Grenade4PistolsSQL == null){

Grenade4PistolsSQL = "HE Grenade";

}

bHEGrenade.setEnabled(false);

grenadeCountPistols = grenadeCountPistols + 1;

sqlStringHEBuyCondition = "True";

if(grenadeCountPistols == 4){

bMolotov.setEnabled(false);

bFlashbang.setEnabled(false);

bSmoke.setEnabled(false);

bDecoy.setEnabled(false);

bIncendiary.setEnabled(false);

sqlStringLimitGrenadeCondition = "True";

}

}

if(source == bSmoke){

intRSBalance = intRSBalance - 300;

tBalance.setText(intRSBalance + "$");

if(Grenade1PistolsSQL == null){

Grenade1PistolsSQL = "Smoke";

}else if(Grenade2PistolsSQL == null){

Grenade2PistolsSQL = "Smoke";

}else if(Grenade3PistolsSQL == null){

Grenade3PistolsSQL = "Smoke";

}else if(Grenade4PistolsSQL == null){

Grenade4PistolsSQL = "Smoke";

}

bSmoke.setEnabled(false);

grenadeCountPistols = grenadeCountPistols + 1;

sqlStringSmokeBuyCondition = "True";

if(grenadeCountPistols == 4){

bMolotov.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bDecoy.setEnabled(false);

bIncendiary.setEnabled(false);

sqlStringLimitGrenadeCondition = "True";

}

}

//Pistols

if(source == bDualBerettas){ //done

intRSBalance = intRSBalance - 300;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "Dual Berettas";

bDualBerettas.setEnabled(false);

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 200){

bGlock18.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TTec9") && intRSBalance >= 500){

bTec9.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(sqlStringCTuspsORp2000.equals("CTUSPS") && intRSBalance >= 200){

bUSPS.setEnabled(true);

}

if(sqlStringCTuspsORp2000.equals("CTP2000") && intRSBalance >= 200){

bP2000.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CT57") && intRSBalance >= 500){

bFiveSeven.setEnabled(true);

}

}

if(intRSBalance >= 300){

bP250.setEnabled(true);

}

}

if (source == bGlock18){ //done

intRSBalance = intRSBalance - 200;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "Glock-18";

bGlock18.setEnabled(false);

if(intRSBalance >= 300){

bP250.setEnabled(true);

bDualBerettas.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TCZ75") == true && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TTec9") == true && intRSBalance >= 500){

bTec9.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(source == bP250){ //done

intRSBalance = intRSBalance - 300;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "P250";

bP250.setEnabled(false);

if(intRSBalance >= 300){

bDualBerettas.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 200){

bGlock18.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TTec9") && intRSBalance >= 500){

bTec9.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(sqlStringCTuspsORp2000.equals("CTUSPS") && intRSBalance >= 200){

bUSPS.setEnabled(true);

}

if(sqlStringCTuspsORp2000.equals("CTP2000") && intRSBalance >= 200){

bP2000.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CT57") && intRSBalance >= 500){

bFiveSeven.setEnabled(true);

}

}

}

if(source == bCZ75Auto){ //done

intRSBalance = intRSBalance - 500;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "CZ75-Auto";

bCZ75Auto.setEnabled(false);

if(intRSBalance >= 300){

bP250.setEnabled(true);

bDualBerettas.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 200){

bGlock18.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(sqlStringCTuspsORp2000.equals("CTUSPS") && intRSBalance >= 200){

bUSPS.setEnabled(true);

}

if(sqlStringCTuspsORp2000.equals("CTP2000") && intRSBalance >= 200){

bP2000.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

}

if(source == bDesertEagle){ //done

intRSBalance = intRSBalance - 700;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "Desert Eagle";

bDesertEagle.setEnabled(false);

if(intRSBalance >= 300){

bP250.setEnabled(true);

bDualBerettas.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 200){

bGlock18.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TTec9") && intRSBalance >= 500){

bTec9.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(sqlStringCTuspsORp2000.equals("CTUSPS") && intRSBalance >= 200){

bUSPS.setEnabled(true);

}

if(sqlStringCTuspsORp2000.equals("CTP2000") && intRSBalance >= 200){

bP2000.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CT57") && intRSBalance >= 500){

bFiveSeven.setEnabled(true);

}

}

}

if(source == bR8Revolver){ //done

intRSBalance = intRSBalance - 600;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "R8 Revolver";

bR8Revolver.setEnabled(false);

if(intRSBalance >= 300){

bDualBerettas.setEnabled(true);

bP250.setEnabled(true);

}

if(sqlStringSide.equals("TSide")){

if(intRSBalance >= 200){

bGlock18.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringTCZ75orTec9.equals("TTec9") && intRSBalance >= 500){

bTec9.setEnabled(true);

}

}

if(sqlStringSide.equals("CTSide")){

if(intRSBalance >= 200){

bUSPS.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CT57") && intRSBalance >= 500){

bFiveSeven.setEnabled(true);

}

}

}

if(source == bTec9){ //done

intRSBalance = intRSBalance - 500;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "Tec-9";

bTec9.setEnabled(false);

if(intRSBalance >= 300){

bDualBerettas.setEnabled(true);

bP250.setEnabled(true);

}

if(intRSBalance >= 200){

bGlock18.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringTDeagleORr8.equals("TR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(source == bFiveSeven){ //done

intRSBalance = intRSBalance - 500;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "Five-SeveN";

bFiveSeven.setEnabled(false);

if(intRSBalance >= 300){

bP250.setEnabled(true);

bDualBerettas.setEnabled(true);

}

if(sqlStringCTuspsORp2000.equals("CTUSPS") && intRSBalance >= 200){

bUSPS.setEnabled(true);

}

if(sqlStringCTuspsORp2000.equals("CTP2000") && intRSBalance >= 200){

bP2000.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(source == bUSPS){ //DONE

intRSBalance = intRSBalance - 200;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "USP-S";

bUSPS.setEnabled(false);

if(intRSBalance >= 300){

bP250.setEnabled(true);

bDualBerettas.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CT57") && intRSBalance >= 500){

bFiveSeven.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

if(source == bP2000){ //DONE

intRSBalance = intRSBalance - 200;

tBalance.setText(intRSBalance + "$");

Weapon1PistolsSQL = "P2000";

bP2000.setEnabled(false);

if(intRSBalance >= 300){

bP250.setEnabled(true);

bDualBerettas.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CTCZ75") && intRSBalance >= 500){

bCZ75Auto.setEnabled(true);

}

if(sqlStringsqlCTCZ75or57.equals("CT57") && intRSBalance >= 500){

bFiveSeven.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTDeagle") && intRSBalance >= 700 ){

bDesertEagle.setEnabled(true);

}

if(sqlStringCTDeagleORr8.equals("CTR8") && intRSBalance >= 600){

bR8Revolver.setEnabled(true);

}

}

//Equipment

if(source == bKevlarVest){

intRSBalance = intRSBalance - 650;

tBalance.setText(intRSBalance + "$");

ArmorPistolsSQL = "Kevlar Vest";

bKevlarVest.setEnabled(false);

bKevlarVestHelmet.setEnabled(false);

playerHasFullArmor = true;

bHelmet.setEnabled(true);

}

if(source == bKevlarVestHelmet){

intRSBalance = intRSBalance - 1000;

tBalance.setText(intRSBalance + "$");

ArmorPistolsSQL = "Kevlar Vest + Helmet";

playerHasFullArmorAndHelmet = true;

bKevlarVestHelmet.setEnabled(false);

bKevlarVest.setEnabled(false);

bHelmet.setEnabled(false);

}

if(source == bHelmet){

intRSBalance = intRSBalance - 350;

tBalance.setText(intRSBalance + "$");

ArmorPistolsSQL = "Kevlar Vest + Helmet";

bHelmet.setEnabled(false);

}

if(source == bZeusx27){

intRSBalance = intRSBalance - 200;

tBalance.setText(intRSBalance + "$");

ZeusPistolsSQL = "Zeus";

bZeusx27.setEnabled(false);

}

if(source == bKits){

intRSBalance = intRSBalance - 400;

tBalance.setText(intRSBalance + "$");

kitsStringPistolsSQL = "Kits";

bKits.setEnabled(false);

}

if (source == bQuitPistolsMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalance+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Armor = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ?, MainGameCondition = ?, PistolsCondition = ?, MolotovBuyCondition = ?, DecoyBuyCondition = ?, FlashBuyCondition = ?, SmokeBuyCondition = ?, HEBuyCondition = ?, LimitGrenadeCondition = ?, PistolNullCondition = ?, kits = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, Weapon1PistolsSQL);

preparedStatement.setString(2, Weapon2PistolsSQL);

preparedStatement.setString(3, ArmorPistolsSQL);

preparedStatement.setString(4, Grenade1PistolsSQL);

preparedStatement.setString(5, Grenade2PistolsSQL);

preparedStatement.setString(6, Grenade3PistolsSQL);

preparedStatement.setString(7, Grenade4PistolsSQL);

preparedStatement.setString(8, ZeusPistolsSQL);

preparedStatement.setString(9, "Game");

preparedStatement.setString(10, "True");

preparedStatement.setString(11, sqlStringMolotovBuyCondition);

preparedStatement.setString(12, sqlStringDecoyBuyCondition);

preparedStatement.setString(13, sqlStringFlashBuyCondition);

preparedStatement.setString(14, sqlStringSmokeBuyCondition);

preparedStatement.setString(15, sqlStringHEBuyCondition);

preparedStatement.setString(16, sqlStringLimitGrenadeCondition);

preparedStatement.setString(17, "True");

preparedStatement.setString(18, kitsStringPistolsSQL);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new BuyMenu();

dispose();

}

//Balance Condition for the buttons [done]

tWeapon1.setText(Weapon1PistolsSQL + ""); //updates the text field, with currently purchased weapon

tWeapon2.setText(Weapon2PistolsSQL + ""); //updates the text field, with currently purchased weapon

tZeus.setText(ZeusPistolsSQL + ""); //updates the text field, with currently purchased weapon

tGrenade1.setText(Grenade1PistolsSQL + ""); //updates the text field, with currently purchased weapon

tGrenade2.setText(Grenade2PistolsSQL + ""); //updates the text field, with currently purchased weapon

tGrenade3.setText(Grenade3PistolsSQL + ""); //updates the text field, with currently purchased weapon

tGrenade4.setText(Grenade4PistolsSQL + ""); //updates the text field, with currently purchased weapon

tKits.setText(kitsStringPistolsSQL + ""); //updates the text field, with currently purchased weapon

tArmor.setText(ArmorPistolsSQL + ""); //updates the text field, with currently purchased weapon

if(intRSBalance < 5200){ //disables the opportunity to buy a weapon, if there is not enough balance

bM249.setEnabled(false);

}

if(intRSBalance < 5000){ //disables the opportunity to buy a weapon, if there is not enough balance

bG3SG1.setEnabled(false);

bSCAR20.setEnabled(false);

}

if(intRSBalance < 4750){ //disables the opportunity to buy a weapon, if there is not enough balance

bAWP.setEnabled(false);

}

if(intRSBalance < 3300){ //disables the opportunity to buy a weapon, if there is not enough balance

bAUG.setEnabled(false);

}

if(intRSBalance < 3100){ //disables the opportunity to buy a weapon, if there is not enough balance

bM4A4.setEnabled(false);

}

if(intRSBalance < 3000){ //disables the opportunity to buy a weapon, if there is not enough balance

bSG553.setEnabled(false);

}

if(intRSBalance < 2900){ //disables the opportunity to buy a weapon, if there is not enough balance

bM4A1S.setEnabled(false);

}

if(intRSBalance < 2700){ //disables the opportunity to buy a weapon, if there is not enough balance

bAK47.setEnabled(false);

}

if(intRSBalance < 2350){ //disables the opportunity to buy a weapon, if there is not enough balance

bP90.setEnabled(false);

}

if(intRSBalance < 2050){ //disables the opportunity to buy a weapon, if there is not enough balance

bFAMAS.setEnabled(false);

}

if(intRSBalance < 2000){ //disables the opportunity to buy a weapon, if there is not enough balance

bXM1014.setEnabled(false);

}

if(intRSBalance < 1800){ //disables the opportunity to buy a weapon, if there is not enough balance

bGalilAR.setEnabled(false);

}

if(intRSBalance < 1700){

bSSG08.setEnabled(false);

bNegev.setEnabled(false);

}

if(intRSBalance < 1500){

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

}

if(intRSBalance < 1400){

bPPBizon.setEnabled(false);

}

if(intRSBalance < 1250){

bMP9.setEnabled(false);

}

if(intRSBalance < 1200){

bUMP45.setEnabled(false);

}

if(intRSBalance < 1100){

bSawedOff.setEnabled(false);

}

if(intRSBalance < 1050){

bMAC10.setEnabled(false);

bNova.setEnabled(false);

}

if(intRSBalance < 1000){

bKevlarVestHelmet.setEnabled(false);

}

if(intRSBalance < 700){

bDesertEagle.setEnabled(false);

}

if(intRSBalance < 650){

bKevlarVest.setEnabled(false);

}

if(intRSBalance < 600){

bR8Revolver.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(intRSBalance < 500){

bCZ75Auto.setEnabled(false);

bTec9.setEnabled(false);

bFiveSeven.setEnabled(false);

}

if(intRSBalance < 400){

bMolotov.setEnabled(false);

bKits.setEnabled(false);

}

if(intRSBalance < 350){

bHelmet.setEnabled(false);

}

if(intRSBalance < 300){

bDualBerettas.setEnabled(false);

bP250.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

}

if(intRSBalance < 200){

bGlock18.setEnabled(false);

bZeusx27.setEnabled(false);

bFlashbang.setEnabled(false);

bP2000.setEnabled(false);

bUSPS.setEnabled(false);

}

if(intRSBalance < 50){

bDecoy.setEnabled(false);

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class buyMenuPickDrop extends EasyApp

{

public static void main(String[] args)

{ new buyMenuPickDrop();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropChoice = addButton("Did you drop a weapon?",100,300,798,100,this);

Button bPickChoice = addButton("Did you pick a weapon?",100,400,798,100,this);

Button bQuitBuyMenu = addButton("Go Back",850,600,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountbuyMenuPickDrop;

int BalancebuyMenuPickDrop;

int RoundCTbuyMenuPickDrop;

int RoundTbuyMenuPickDrop;

public buyMenuPickDrop(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountbuyMenuPickDrop = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountbuyMenuPickDrop.next() ){

String sqlRoundCountstringbuyMenuPickDrop = sqlRoundCountbuyMenuPickDrop.getString(1);

RoundCountbuyMenuPickDrop = Integer.parseInt (sqlRoundCountstringbuyMenuPickDrop);}

ResultSet rsBalancebuyMenuPickDrop = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancebuyMenuPickDrop.next() ){

String rs1BalancestringbuyMenuPickDrop = rsBalancebuyMenuPickDrop.getString(1);

BalancebuyMenuPickDrop = Integer.parseInt (rs1BalancestringbuyMenuPickDrop);}

ResultSet sqlRoundCTbuyMenuPickDrop = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTbuyMenuPickDrop.next() ){

String sqlRoundCTstringbuyMenuPickDrop = sqlRoundCTbuyMenuPickDrop.getString(1);

RoundCTbuyMenuPickDrop = Integer.parseInt (sqlRoundCTstringbuyMenuPickDrop);}

ResultSet sqlRoundTbuyMenuPickDrop = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTbuyMenuPickDrop .next() ){

String sqlRoundTstringbuyMenuPickDrop = sqlRoundTbuyMenuPickDrop .getString(1);

RoundTbuyMenuPickDrop = Integer.parseInt (sqlRoundTstringbuyMenuPickDrop);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,700);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountbuyMenuPickDrop + "");

tBalance.setText(BalancebuyMenuPickDrop + "$");

tCTscore.setText(RoundCTbuyMenuPickDrop + "");

tTscore.setText(RoundTbuyMenuPickDrop + "");

}

public void actions(Object source,String command){

if (source == bQuitBuyMenu)

{

new BuyMenu();

dispose();

}

if (source == bDropChoice)

{

new BuyMenuDropMenu();

dispose();

}

if (source == bPickChoice)

{

new BuyMenuPickMenu();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class BuyMenuDropMenu extends EasyApp

{

public static void main(String[] args)

{ new BuyMenuDropMenu();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,800,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropWeapon1 = addButton("Secondary weapon",400,100,183,100,this);

Button bDropWeapon2 = addButton("Primary weapon",400,200,183,100,this);

Button bDropZeus = addButton("Zeus",400,300,183,100,this);

Button bDropGrenade1 = addButton("Grenade 1",400,400,183,100,this);

Button bDropGrenade2 = addButton("Grenade 2",400,500,183,100,this);

Button bDropGrenade3 = addButton("Grenade 3",400,600,183,100,this);

Button bDropGrenade4 = addButton("Grenade 4",400,700,183,100,this);

Button bQuitBuyMenuDropMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,800,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lDropWeapon1 = addLabel("",227,100,100,30,this);

Label lDropWeapon2 = addLabel("",227,200,100,30,this);

Label lDropZeus = addLabel("",227,300,100,30,this);

Label lDropGrenade1 = addLabel("",227,400,100,30,this);

Label lDropGrenade2 = addLabel("",227,500,100,30,this);

Label lDropGrenade3 = addLabel("",227,600,100,30,this);

Label lDropGrenade4 = addLabel("",227,700,100,30,this);

int RoundCountbuyMenuDrop;

int BalancebuyMenuDrop;

int RoundCTbuyMenuDrop;

int RoundTbuyMenuDrop;

String buyMenuDropWeapon1;

String buyMenuDropWeapon2;

String buyMenuDropZeus;

String buyMenuDropGrenade1;

String buyMenuDropGrenade2;

String buyMenuDropGrenade3;

String buyMenuDropGrenade4;

public BuyMenuDropMenu(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountBuyMenuDropMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountBuyMenuDropMenu.next() ){

String sqlRoundCountstringBuyMenuDropMenu = sqlRoundCountBuyMenuDropMenu.getString(1);

RoundCountbuyMenuDrop = Integer.parseInt (sqlRoundCountstringBuyMenuDropMenu);}

ResultSet rsBalanceBuyMenuDropMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalanceBuyMenuDropMenu.next() ){

String rs1BalancestringBuyMenuDropMenu = rsBalanceBuyMenuDropMenu.getString(1);

BalancebuyMenuDrop = Integer.parseInt (rs1BalancestringBuyMenuDropMenu);}

ResultSet sqlRoundCTBuyMenuDropMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTBuyMenuDropMenu.next() ){

String sqlRoundCTstringBuyMenuDropMenu = sqlRoundCTBuyMenuDropMenu.getString(1);

RoundCTbuyMenuDrop = Integer.parseInt (sqlRoundCTstringBuyMenuDropMenu);}

ResultSet sqlRoundTBuyMenuDropMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTBuyMenuDropMenu .next() ){

String sqlRoundTstringBuyMenuDropMenu = sqlRoundTBuyMenuDropMenu .getString(1);

RoundTbuyMenuDrop = Integer.parseInt (sqlRoundTstringBuyMenuDropMenu);}

ResultSet sqlWeapon1BuyMenuDropMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1BuyMenuDropMenu .next() ){

buyMenuDropWeapon1 = sqlWeapon1BuyMenuDropMenu .getString(1);

}

ResultSet sqlWeapon2BuyMenuDropMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2BuyMenuDropMenu .next() ){

buyMenuDropWeapon2 = sqlWeapon2BuyMenuDropMenu .getString(1);

}

ResultSet sqlZeusBuyMenuDropMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeusBuyMenuDropMenu .next() ){

buyMenuDropZeus = sqlZeusBuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade1BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1BuyMenuDropMenu .next() ){

buyMenuDropGrenade1 = sqlGrenade1BuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade2BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2BuyMenuDropMenu .next() ){

buyMenuDropGrenade2 = sqlGrenade2BuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade3BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3BuyMenuDropMenu .next() ){

buyMenuDropGrenade3 = sqlGrenade3BuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade4BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4BuyMenuDropMenu .next() ){

buyMenuDropGrenade4 = sqlGrenade4BuyMenuDropMenu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountbuyMenuDrop + "");

tBalance.setText(BalancebuyMenuDrop + "$");

tCTscore.setText(RoundCTbuyMenuDrop + "");

tTscore.setText(RoundTbuyMenuDrop + "");

lDropWeapon1.setText(buyMenuDropWeapon1);

lDropWeapon2.setText(buyMenuDropWeapon2);

lDropZeus.setText(buyMenuDropZeus);

lDropGrenade1.setText(buyMenuDropGrenade1);

lDropGrenade2.setText(buyMenuDropGrenade2);

lDropGrenade3.setText(buyMenuDropGrenade3);

lDropGrenade4.setText(buyMenuDropGrenade4);

if(buyMenuDropWeapon1 == null){

bDropWeapon1.setVisible(false);

}

if(buyMenuDropWeapon2 == null){

bDropWeapon2.setVisible(false);

}

if(buyMenuDropZeus == null){

bDropZeus.setVisible(false);

}

if(buyMenuDropGrenade1 == null){

bDropGrenade1.setVisible(false);

}

if(buyMenuDropGrenade2 == null){

bDropGrenade2.setVisible(false);

}

if(buyMenuDropGrenade3 == null){

bDropGrenade3.setVisible(false);

}

if(buyMenuDropGrenade4 == null){

bDropGrenade4.setVisible(false);

}

}

public void actions(Object source,String command){

if (source == bDropWeapon1){

bDropWeapon1.setEnabled(false);

buyMenuDropWeapon1 = null;

}

if (source == bDropWeapon2){

bDropWeapon2.setEnabled(false);

buyMenuDropWeapon2 = null;

}

if (source == bDropZeus){

bDropZeus.setEnabled(false);

buyMenuDropZeus = null;

}

if (source == bDropGrenade1){

bDropGrenade1.setEnabled(false);

buyMenuDropGrenade1 = null;

}

if (source == bDropGrenade2){

bDropGrenade2.setEnabled(false);

buyMenuDropGrenade2 = null;

}

if (source == bDropGrenade3){

bDropGrenade3.setEnabled(false);

buyMenuDropGrenade3 = null;

}

if (source == bDropGrenade4){

bDropGrenade4.setEnabled(false);

buyMenuDropGrenade4 = null;

}

if (source == bQuitBuyMenuDropMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,buyMenuDropWeapon1);

preparedStatement.setString(2,buyMenuDropWeapon2);

preparedStatement.setString(3,buyMenuDropGrenade1);

preparedStatement.setString(4,buyMenuDropGrenade2);

preparedStatement.setString(5,buyMenuDropGrenade3);

preparedStatement.setString(6,buyMenuDropGrenade4);

preparedStatement.setString(7,buyMenuDropZeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new buyMenuPickDrop();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class BuyMenuPickMenu extends EasyApp

{

public static void main(String[] args)

{ new BuyMenuPickMenu();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

TextField tWeapon1 = addTextField("",370,600,150,30,this);

TextField tWeapon2 = addTextField("",370,635,150,30,this);

TextField tZeus = addTextField("",370,670,150,30,this);

TextField tGrenade1 = addTextField("",370,705,150,30,this);

TextField tGrenade2 = addTextField("",370,740,150,30,this);

TextField tGrenade3 = addTextField("",370,775,150,30,this);

TextField tGrenade4 = addTextField("",370,810,150,30,this);

TextField tKits = addTextField("",370,845,150,30,this);

Label lGrenadeLabelPickMenu = addLabel("Grenades",70,90,60,30,this);

Label lPistolsLabelPickMenu = addLabel("Pistols",254,90,60,30,this);

Label lZeusLabelPickMenu = addLabel("Equipment",476,90,60,30,this);

Label lSMGLabelPickMenu = addLabel("SMG",742,90,60,30,this);

Label lRiflesLabelPickMenu = addLabel("Rifles",1008,90,60,30,this);

Label lHeavyLabelPickMenu = addLabel("Heavy",1274,90,60,30,this);

Label lYourEquipment = addLabel("Your equipment:",370,570,150,30,this);

Button bMolotov = addButton("Molotov",0,120,183,100, this);

Button bDecoy = addButton("Decoy",0,220,183,100, this);

Button bFlashbang = addButton("Flashbang",0,320,183,100,this);

Button bHEGrenade = addButton("HE Grenade",0,420,183,100,this);

Button bSmoke = addButton("Smoke",0,520,183,100,this);

Button bIncendiary = addButton("Incendiary Grenade",0,620,183,100,this);

Button bGlock18 = addButton("Glock-18",183,120,183,50,this);

Button bDualBerettas= addButton("Dual Berettas",183,170,183,50,this);

Button bP250 = addButton("P250",183,220,183,50,this);

Button bCZ75Auto = addButton("CZ75 - Auto",183,270,183,50,this);

Button bDesertEagle = addButton("Desert Eagle",183,320,183,50,this);

Button bR8Revolver = addButton("R8 Revolver",183,370,183,50,this);

Button bTec9 = addButton("Tec-9",183,420,183,50,this);

Button bUSPS = addButton("USP-S",183,470,183,50,this);

Button bP2000 = addButton("P2000",183,520,183,50,this);

Button bFiveSeveN = addButton("Five-SeveN",183,570,183,50,this);

Button bZeusx27 = addButton("Zeus x27",366,120,266,50,this);

Button bKits = addButton("Kits",366,170,266,50,this);

Button bMAC10 = addButton("MAC-10",632,120,266,50,this);

Button bMP5SD = addButton("MP5-SD",632,170,266,50,this);

Button bMP7 = addButton("MP7",632,220,266,50,this);

Button bUMP45 = addButton("UMP - 45",632,270,266,50,this);

Button bP90 = addButton("P90",632,320,266,50,this);

Button bPPBizon = addButton("PP-Bizon",632,370,266,50,this);

Button bMP9 = addButton("MP9",632,420,266,50,this);

Button bGalilAR = addButton("Galil AR",898,120,266,50,this);

Button bAK47 = addButton("AK-47",898,170,266,50,this);

Button bSSG08 = addButton("SSG 08",898,220,266,50,this);

Button bSG553 = addButton("SG 553",898,270,266,50,this);

Button bAWP = addButton("AWP",898,320,266,50,this);

Button bG3SG1 = addButton("G3SG1",898,370,266,50,this);

Button bFAMAS = addButton("FAMAS",898,420,266,50,this);

Button bM4A4 = addButton("M4A4",898,470,266,50,this);

Button bM4A1S = addButton("M4A1-S",898,520,266,50,this);

Button bAUG = addButton("AUG",898,570,266,50,this);

Button bSCAR20 = addButton("SCAR-20",898,620,266,50,this);

Button bNova = addButton("Nova",1164,120,266,50,this);

Button bXM1014 = addButton("XM1014",1164,170,266,50,this);

Button bSawedOff = addButton("Sawed-Off",1164,220,266,50,this);

Button bM249 = addButton("M249",1164,270,266,50,this);

Button bNegev= addButton("Negev",1164,320,266,50,this);

Button bMAG7= addButton("MAG-7",1164,370,266,50,this);

Button bQuitbuyMenuPickMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,20,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountbuyMenuPick;

int BalancebuyMenuPick;

int RoundCTbuyMenuPick;

int RoundTbuyMenuPick;

int flashbangCountPick = 0;

String buyMenuPickWeapon1;

String buyMenuPickWeapon2;

String buyMenuPickZeus;

String buyMenuPickGrenade1;

String buyMenuPickGrenade2;

String buyMenuPickGrenade3;

String buyMenuPickGrenade4;

String buyMenuPickKits;

String sqlStringLimitGrenadeConditionPickMenu;

String sqlStringSidePickMenu;

public BuyMenuPickMenu(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountbuyMenuPickMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountbuyMenuPickMenu.next() ){

String sqlRoundCountstringbuyMenuPickMenu = sqlRoundCountbuyMenuPickMenu.getString(1);

RoundCountbuyMenuPick = Integer.parseInt (sqlRoundCountstringbuyMenuPickMenu);}

ResultSet rsBalancebuyMenuPickMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancebuyMenuPickMenu.next() ){

String rs1BalancestringbuyMenuPickMenu = rsBalancebuyMenuPickMenu.getString(1);

BalancebuyMenuPick = Integer.parseInt (rs1BalancestringbuyMenuPickMenu);}

ResultSet sqlRoundCTbuyMenuPickMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTbuyMenuPickMenu.next() ){

String sqlRoundCTstringbuyMenuPickMenu = sqlRoundCTbuyMenuPickMenu.getString(1);

RoundCTbuyMenuPick = Integer.parseInt (sqlRoundCTstringbuyMenuPickMenu);}

ResultSet sqlRoundTbuyMenuPickMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTbuyMenuPickMenu .next() ){

String sqlRoundTstringbuyMenuPickMenu = sqlRoundTbuyMenuPickMenu .getString(1);

RoundTbuyMenuPick = Integer.parseInt (sqlRoundTstringbuyMenuPickMenu);}

ResultSet sqlWeapon1buyMenuPickMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1buyMenuPickMenu .next() ){

buyMenuPickWeapon1 = sqlWeapon1buyMenuPickMenu .getString(1);

}

ResultSet sqlWeapon2buyMenuPickMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2buyMenuPickMenu .next() ){

buyMenuPickWeapon2 = sqlWeapon2buyMenuPickMenu .getString(1);

}

ResultSet sqlZeusbuyMenuPickMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeusbuyMenuPickMenu .next() ){

buyMenuPickZeus = sqlZeusbuyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade1buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1buyMenuPickMenu .next() ){

buyMenuPickGrenade1 = sqlGrenade1buyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade2buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2buyMenuPickMenu .next() ){

buyMenuPickGrenade2 = sqlGrenade2buyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade3buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3buyMenuPickMenu .next() ){

buyMenuPickGrenade3 = sqlGrenade3buyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade4buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4buyMenuPickMenu .next() ){

buyMenuPickGrenade4 = sqlGrenade4buyMenuPickMenu .getString(1);

}

ResultSet sqlLimitGrenadeConditionPickMenu = stmt.executeQuery( "SELECT LimitGrenadeCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlLimitGrenadeConditionPickMenu.next() ){

sqlStringLimitGrenadeConditionPickMenu = sqlLimitGrenadeConditionPickMenu.getString(1);}

ResultSet sqlSidePickMenu = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlSidePickMenu.next() ){

sqlStringSidePickMenu = sqlSidePickMenu.getString(1);}

ResultSet kitsPistolsSQL = stmt.executeQuery( "SELECT kits FROM IACompCSGO WHERE idNumCS = 1;") ;

while( kitsPistolsSQL.next() ){

buyMenuPickKits = kitsPistolsSQL.getString(1);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountbuyMenuPick + "");

tBalance.setText(BalancebuyMenuPick + "$");

tCTscore.setText(RoundCTbuyMenuPick + "");

tTscore.setText(RoundTbuyMenuPick + "");

tWeapon1.setText(buyMenuPickWeapon1 + "");

tWeapon2.setText(buyMenuPickWeapon2 + "");

tZeus.setText(buyMenuPickZeus + "");

tGrenade1.setText(buyMenuPickGrenade1 + "");

tGrenade2.setText(buyMenuPickGrenade2 + "");

tGrenade3.setText(buyMenuPickGrenade3 + "");

tGrenade4.setText(buyMenuPickGrenade4 + "");

tKits.setText(buyMenuPickKits + "");

if((RoundCountbuyMenuPick == 1 || RoundCountbuyMenuPick == 16) && sqlStringSidePickMenu.equals("TSide")){

bIncendiary.setEnabled(false);

bUSPS.setEnabled(false);

bP2000.setEnabled(false);

bFiveSeveN.setEnabled(false);

bMP9.setEnabled(false);

bFAMAS.setEnabled(false);

bM4A4.setEnabled(false);

bM4A1S.setEnabled(false);

bAUG.setEnabled(false);

bSCAR20.setEnabled(false);

bMAG7.setEnabled(false);

bMAC10.setEnabled(false);

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

bUMP45.setEnabled(false);

bP90.setEnabled(false);

bPPBizon.setEnabled(false);

bMP9.setEnabled(false);

bGalilAR.setEnabled(false);

bAK47.setEnabled(false);

bSSG08.setEnabled(false);

bSG553.setEnabled(false);

bAWP.setEnabled(false);

bG3SG1.setEnabled(false);

bNova.setEnabled(false);

bXM1014.setEnabled(false);

bSawedOff.setEnabled(false);

bM249.setEnabled(false);

bNegev.setEnabled(false);

}

if((RoundCountbuyMenuPick == 1 || RoundCountbuyMenuPick == 16) && sqlStringSidePickMenu.equals("CTSide")){

bMolotov.setEnabled(false);

bGlock18.setEnabled(false);

bTec9.setEnabled(false);

bMP9.setEnabled(false);

bFAMAS.setEnabled(false);

bM4A4.setEnabled(false);

bM4A1S.setEnabled(false);

bAUG.setEnabled(false);

bSCAR20.setEnabled(false);

bMAG7.setEnabled(false);

bMAC10.setEnabled(false);

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

bUMP45.setEnabled(false);

bP90.setEnabled(false);

bPPBizon.setEnabled(false);

bMP9.setEnabled(false);

bGalilAR.setEnabled(false);

bAK47.setEnabled(false);

bSSG08.setEnabled(false);

bSG553.setEnabled(false);

bAWP.setEnabled(false);

bG3SG1.setEnabled(false);

bNova.setEnabled(false);

bXM1014.setEnabled(false);

bSawedOff.setEnabled(false);

bM249.setEnabled(false);

bNegev.setEnabled(false);

}

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Flashbang")){

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade1.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Flashbang")){

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade2.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Flashbang")){

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade3.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Flashbang")){

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade4.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickKits != null){

if(buyMenuPickKits.equals("Kits")){

bKits.setEnabled(false);

}

}

if(sqlStringSidePickMenu.equals("TSide")){

bKits.setVisible(false);

}

}

public void actions(Object source,String command){

//grenades

if(source == bMolotov){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Molotov";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Molotov";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Molotov";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Molotov";

tGrenade4.setText(buyMenuPickGrenade4);

}

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bDecoy){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Decoy";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Decoy";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Decoy";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Decoy";

tGrenade4.setText(buyMenuPickGrenade4);

}

bDecoy.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bFlashbang){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Flashbang";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Flashbang";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Flashbang";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Flashbang";

tGrenade4.setText(buyMenuPickGrenade4);

}

flashbangCountPick = flashbangCountPick + 1;

if(flashbangCountPick == 2){

bFlashbang.setEnabled(false);

}

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bHEGrenade){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "HE Grenade";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "HE Grenade";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "HE Grenade";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "HE Grenade";

tGrenade4.setText(buyMenuPickGrenade4);

}

bHEGrenade.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bSmoke){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Smoke";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Smoke";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Smoke";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Smoke";

tGrenade4.setText(buyMenuPickGrenade4);

}

bSmoke.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bIncendiary){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Incendiary Grenade";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Incendiary Grenade";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Incendiary Grenade";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Incendiary Grenade";

tGrenade4.setText(buyMenuPickGrenade4);

}

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

//pistols

if (source == bGlock18){

buyMenuPickWeapon1 = "Glock-18";

tWeapon1.setText(buyMenuPickWeapon1);

}

if (source == bDualBerettas){

buyMenuPickWeapon1 = "Dual Berettas";

tWeapon1.setText(buyMenuPickWeapon1);

}

if (source == bP250){

buyMenuPickWeapon1 = "P250";

tWeapon1.setText(buyMenuPickWeapon1);

}

if (source == bCZ75Auto){

buyMenuPickWeapon1 = "CZ75-Auto";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bDesertEagle){

buyMenuPickWeapon1 = "Desert Eagle";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bR8Revolver){

buyMenuPickWeapon1 = "R8 Revolver";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bTec9 ){

buyMenuPickWeapon1 = "Tec-9";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bUSPS){

buyMenuPickWeapon1 = "USP-S";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bP2000){

buyMenuPickWeapon1 = "P2000";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bFiveSeveN){

buyMenuPickWeapon1 = "Five-SeveN";

tWeapon1.setText(buyMenuPickWeapon1);

}

//zeus

if(source == bZeusx27){

buyMenuPickZeus = "Zeus";

tZeus.setText(buyMenuPickZeus);

}

//for kits

if(source == bKits){

bKits.setEnabled(false);

buyMenuPickKits = "Kits";

tKits.setText(buyMenuPickKits);

}

//smgs

if(source == bMAC10){

buyMenuPickWeapon2 = "MAC-10";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMP5SD){

buyMenuPickWeapon2 = "MP5-SD";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMP7){

buyMenuPickWeapon2 = "MP7";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bUMP45){

buyMenuPickWeapon2 = "UMP-45";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bP90){

buyMenuPickWeapon2 = "P90";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bPPBizon){

buyMenuPickWeapon2 = "PP-Bizon";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMP9){

buyMenuPickWeapon2 = "MP9";

tWeapon2.setText(buyMenuPickWeapon2);

}

//rifles

if(source == bGalilAR){

buyMenuPickWeapon2 = "Galil AR";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bAK47){

buyMenuPickWeapon2 = "AK-47";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSSG08){

buyMenuPickWeapon2 = "SSG 08";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSG553){

buyMenuPickWeapon2 = "SG 553";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bAWP){

buyMenuPickWeapon2 = "AWP";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bG3SG1){

buyMenuPickWeapon2 = "G3SG1";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bFAMAS){

buyMenuPickWeapon2 = "FAMAS";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bM4A4){

buyMenuPickWeapon2 = "M4A4";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bM4A1S){

buyMenuPickWeapon2 = "M4A1-S";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bAUG){

buyMenuPickWeapon2 = "AUG";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSCAR20){

buyMenuPickWeapon2 = "SCAR-20";

tWeapon2.setText(buyMenuPickWeapon2);

}

//heavy

if(source == bNova){

buyMenuPickWeapon2 = "Nova";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bXM1014){

buyMenuPickWeapon2 = "XM1014";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSawedOff){

buyMenuPickWeapon2 = "Sawed-Off";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bM249){

buyMenuPickWeapon2 = "M249";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bNegev){

buyMenuPickWeapon2 = "Negev";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMAG7){

buyMenuPickWeapon2 = "MAG-7";

tWeapon2.setText(buyMenuPickWeapon2);

}

if (source == bQuitbuyMenuPickMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ?, kits = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,buyMenuPickWeapon1);

preparedStatement.setString(2,buyMenuPickWeapon2);

preparedStatement.setString(3,buyMenuPickGrenade1);

preparedStatement.setString(4,buyMenuPickGrenade2);

preparedStatement.setString(5,buyMenuPickGrenade3);

preparedStatement.setString(6,buyMenuPickGrenade4);

preparedStatement.setString(7,buyMenuPickZeus);

preparedStatement.setString(8,buyMenuPickKits);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new buyMenuPickDrop();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class GameT extends EasyApp

{

public static void main(String[] args)

{

new GameT();

}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bHowManyKills = addButton("How many kills?",0,120,220,100, this);

Button bKillMenu = addButton("Kill Menu",0,220,220,100, this);

Button bBombPlant = addButton("Did you plant the bomb?",0,320,220,100, this);

Button bDropPick = addButton("Did you pick/drop the weapon?",0,420,220,100,this);

Button bGrenadeUse = addButton("Did you use grenades?",0,520,220,100,this);

Button bSuicideReward = addButton("Did you get a suicide reward?",0,620,220,100,this);

Button bTeamKill = addButton("Did you make a team kill?",0,720,220,100,this);

Button bGrenade1 = addButton("Grenade 1",400,120,220,100, this);

Button bGrenade2 = addButton("Grenade 2",400,220,220,100, this);

Button bGrenade3 = addButton("Grenade 3",400,320,220,100, this);

Button bGrenade4 = addButton("Grenade 4",400,420,220,100, this);

Button bFinishGrenadeUse = addButton("Finish with grenades",640,220,220,100, this);

Button bQuitbuyMenuGameTMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lGrenade1 = addLabel("",250,120,100,30,this);

Label lGrenade2 = addLabel("",250,220,100,30,this);

Label lGrenade3 = addLabel("",250,320,100,30,this);

Label lGrenade4 = addLabel("",250,420,100,30,this);

int intRSBalanceGameT;

int intSQLRoundCTGameT;

int intSQLRoundTGameT;

int intSQLRoundNumberGameT;

String sqlHMKconditionString;

String BombPlantString;

String GameTGrenade1;

String GameTGrenade2;

String GameTGrenade3;

String GameTGrenade4;

String buttonBombPlantString;

String buttonSuicideRewardString;

String buttonGrenadeUseString;

String buttonTeamKillString;

String GameTMenuWeapon1;

String GameTMenuWeapon2;

int killNumber;

boolean BombPlantButtonBoolean = false;

boolean SuicideRewardButtonBoolean = false;

boolean TeamKillButtonBoolean = false;

boolean GrenadeUseButtonBoolean = false;

String BombPlantButtonString = "False";

String SuicideRewardButtonString = "False";

String TeamKillButtonString = "False";

String GrenadeUseButtonString = "False";

String sqlStringSide;

public GameT() // Constructor - change window appearance

{

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCount = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCount.next() ){

String sqlRoundCountstring = sqlRoundCount.getString(1);

intSQLRoundNumberGameT = Integer.parseInt (sqlRoundCountstring);}

ResultSet rsBalance = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalance.next() ){

String rs1Balancestring = rsBalance.getString(1);

intRSBalanceGameT = Integer.parseInt (rs1Balancestring);}

ResultSet sqlRoundCT = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCT.next() ){

String sqlRoundCTstring = sqlRoundCT.getString(1);

intSQLRoundCTGameT = Integer.parseInt (sqlRoundCTstring);}

ResultSet sqlRoundT = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundT.next() ){

String sqlRoundTstring = sqlRoundT.getString(1);

intSQLRoundTGameT = Integer.parseInt (sqlRoundTstring);}

ResultSet sqlHMKcondition = stmt.executeQuery( "SELECT HMKcondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlHMKcondition.next() ){

sqlHMKconditionString = sqlHMKcondition.getString(1);}

ResultSet sqlGrenade1GameT = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1GameT .next() ){

GameTGrenade1 = sqlGrenade1GameT .getString(1);

}

ResultSet sqlGrenade2GameT = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2GameT .next() ){

GameTGrenade2 = sqlGrenade2GameT .getString(1);

}

ResultSet sqlGrenade3GameT = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3GameT .next() ){

GameTGrenade3 = sqlGrenade3GameT .getString(1);

}

ResultSet sqlGrenade4GameT = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4GameT .next() ){

GameTGrenade4 = sqlGrenade4GameT .getString(1);

}

ResultSet sqlbuttonBombPlant = stmt.executeQuery( "SELECT buttonBombPlant FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlbuttonBombPlant.next() ){

buttonBombPlantString = sqlbuttonBombPlant.getString(1);

}

ResultSet sqlbuttonSuicideReward = stmt.executeQuery( "SELECT buttonSuicideReward FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlbuttonSuicideReward.next() ){

buttonSuicideRewardString = sqlbuttonSuicideReward.getString(1);

}

ResultSet sqlbuttonGrenadeUse = stmt.executeQuery( "SELECT buttonGrenadeUse FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlbuttonGrenadeUse.next() ){

buttonGrenadeUseString = sqlbuttonGrenadeUse.getString(1);

}

ResultSet sqlbuttonTeamKill = stmt.executeQuery( "SELECT buttonTeamKill FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlbuttonTeamKill.next() ){

buttonTeamKillString = sqlbuttonTeamKill.getString(1);

}

ResultSet sqlGameTMenuWeapon1 = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGameTMenuWeapon1 .next() ){

GameTMenuWeapon1 = sqlGameTMenuWeapon1.getString(1);

}

ResultSet sqlGameTMenuWeapon2 = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGameTMenuWeapon2 .next() ){

GameTMenuWeapon2 = sqlGameTMenuWeapon2.getString(1);

}

ResultSet sqlSide = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlSide.next() ){

sqlStringSide = sqlSide.getString(1);}

//finish it

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(intSQLRoundNumberGameT + "");

tBalance.setText(intRSBalanceGameT + "$");

tCTscore.setText(intSQLRoundCTGameT + "");

tTscore.setText(intSQLRoundTGameT+ "");

bGrenade1.setVisible(false);

bGrenade2.setVisible(false);

bGrenade3.setVisible(false);

bGrenade4.setVisible(false);

bFinishGrenadeUse.setVisible(false);

if(sqlStringSide.equals("CTSide")){

bBombPlant.setVisible(false);

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET buttonBombPlant = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

BombPlantButtonBoolean = true;

BombPlantButtonString = "True";

tBalance.setText(intRSBalanceGameT + "$");

bBombPlant.setEnabled(false);

}

if(buttonBombPlantString.equals("True")){

bBombPlant.setEnabled(false);

}

if(buttonSuicideRewardString.equals("True")){

bSuicideReward.setEnabled(false);

}

if(buttonGrenadeUseString.equals("True")){

bGrenadeUse.setEnabled(false);

}

if(buttonTeamKillString.equals("True")){

bTeamKill.setEnabled(false);

}

if(sqlHMKconditionString.equals("False")){

bBombPlant.setEnabled(false);

bKillMenu.setEnabled(false);

bDropPick.setEnabled(false);

bGrenadeUse.setEnabled(false);

bSuicideReward.setEnabled(false);

bQuitbuyMenuGameTMenu.setEnabled(false);

bTeamKill.setEnabled(false);

}else if(sqlHMKconditionString.equals("True")){

bHowManyKills.setEnabled(false);

bKillMenu.setEnabled(false);

bQuitbuyMenuGameTMenu.setEnabled(false);

}

if(buttonTeamKillString.equals("True") && buttonSuicideRewardString.equals("True") && buttonBombPlantString.equals("True") && buttonGrenadeUseString.equals("True")){

bQuitbuyMenuGameTMenu.setEnabled(true);

}

}

public void actions(Object source,String command){

if(source == bHowManyKills){ //the button to enter the number of kills done in the round

boolean howManyKillsCondition = false; //the condition for the loop

while(howManyKillsCondition == false){ //run the loop until the condition is met

int killNumberHowManyKills = inputInt("How many kills have you done? 0-5"); //calls an input field

if(killNumberHowManyKills == 0){

howManyKillsCondition = true;

bHowManyKills.setEnabled(false);

bBombPlant.setEnabled(true);

bDropPick.setEnabled(true);

bGrenadeUse.setEnabled(true);

bSuicideReward.setEnabled(true);

bTeamKill.setEnabled(true);

killNumber = 0;

sqlHMKconditionString = "True";

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int killNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillNumber = "+killNumber+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET HMKcondition = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, sqlHMKconditionString);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

else if(killNumberHowManyKills == 1){

howManyKillsCondition = true;

bHowManyKills.setEnabled(false);

bKillMenu.setEnabled(true);

killNumber = 1;

}

else if(killNumberHowManyKills == 2){

howManyKillsCondition = true;

bHowManyKills.setEnabled(false);

bKillMenu.setEnabled(true);

killNumber = 2;

}

else if(killNumberHowManyKills == 3){

howManyKillsCondition = true;

bHowManyKills.setEnabled(false);

bKillMenu.setEnabled(true);

killNumber = 3;

}

else if(killNumberHowManyKills == 4){

howManyKillsCondition = true;

bHowManyKills.setEnabled(false);

bKillMenu.setEnabled(true);

killNumber = 4;

}

else if(killNumberHowManyKills == 5){

howManyKillsCondition = true;

bHowManyKills.setEnabled(false);

bKillMenu.setEnabled(true);

killNumber = 5;

}else{

howManyKillsCondition = false;

}

}

}

if(source == bKillMenu){

sqlHMKconditionString = "True";

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int killNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillNumber = "+killNumber+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET HMKcondition = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, sqlHMKconditionString);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuT();

dispose();

}

if(source == bBombPlant) {

boolean BombPlantBoolean = false;

while(BombPlantBoolean == false){

String bombPlantQuestion = inputString("Did you plant the bomb? Y/N");

if(bombPlantQuestion.equals("Y")){

intRSBalanceGameT = intRSBalanceGameT + 300;

if (intRSBalanceGameT > 16000){

intRSBalanceGameT = 16000;

}

BombPlantString = "True";

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceGameT+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET BombPlant = ?, buttonBombPlant = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, BombPlantString);

preparedStatement.setString(2, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

BombPlantBoolean = true;

}else if(bombPlantQuestion.equals("N")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET buttonBombPlant = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

BombPlantBoolean = true;

}else{

BombPlantBoolean = false;

}

}

BombPlantButtonBoolean = true;

BombPlantButtonString = "True";

tBalance.setText(intRSBalanceGameT + "$");

bBombPlant.setEnabled(false);

}

if(source == bGrenade1){

GameTGrenade1 = null;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade1 = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, GameTGrenade1);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bGrenade1.setEnabled(false);

}

if(source == bGrenade2){

GameTGrenade2 = null;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade2 = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, GameTGrenade2);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bGrenade2.setEnabled(false);

}

if(source == bGrenade3){

GameTGrenade3 = null;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade3 = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, GameTGrenade3);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bGrenade3.setEnabled(false);

}

if(source == bGrenade4){

GameTGrenade4 = null;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade4 = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, GameTGrenade4);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bGrenade4.setEnabled(false);

}

if(source == bSuicideReward){

boolean suicideReward = false;

while(suicideReward == false){

String suicideRewardString = inputString("Did you get a suicide reward? Y/N");

if(suicideRewardString.equals("Y")){

boolean whichSuicideReward = false;

while(whichSuicideReward == false){

int WhichSuicideReward = inputInt("Which suicide reward did you get? 100/300/600/900");

if(WhichSuicideReward == 100){

intRSBalanceGameT = intRSBalanceGameT + 100;

whichSuicideReward = true;

}else if(WhichSuicideReward == 300){

intRSBalanceGameT = intRSBalanceGameT + 300;

whichSuicideReward = true;

}else if(WhichSuicideReward == 600){

intRSBalanceGameT = intRSBalanceGameT + 600;

whichSuicideReward = true;

}else if(WhichSuicideReward == 900){

intRSBalanceGameT = intRSBalanceGameT + 900;

whichSuicideReward = true;

}else{

whichSuicideReward = false;

}

}

suicideReward = true;

}else if(suicideRewardString.equals("N")){

suicideReward = true;

}else{

suicideReward = false;

}

}

tBalance.setText(intRSBalanceGameT + "$");

bSuicideReward.setEnabled(false);

SuicideRewardButtonBoolean = true;

SuicideRewardButtonString = "True";

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balance1Update = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceGameT+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET buttonSuicideReward = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(source == bTeamKill){

boolean bTeamKillBoolean = false;

while(bTeamKillBoolean == false){

String bTeamKillString = inputString("Did you make a team kill? Y/N");

if(bTeamKillString.equals("Y")){

boolean bTeamKillString1 = false;

while(bTeamKillString1 == false){

int bTeamKillString1INT = inputInt("How many teamkills did you make? 1-4");

if(bTeamKillString1INT == 1){

intRSBalanceGameT = intRSBalanceGameT - 300;

bTeamKillString1 = true;

}else if(bTeamKillString1INT == 2){

intRSBalanceGameT = intRSBalanceGameT - 600;

bTeamKillString1 = true;

}else if(bTeamKillString1INT == 3){

intRSBalanceGameT = intRSBalanceGameT - 900;

bTeamKillString1 = true;

}else if(bTeamKillString1INT == 4){

intRSBalanceGameT = intRSBalanceGameT - 1200;

bTeamKillString1 = true;

}else{

bTeamKillString1 = false;

}

}

bTeamKillBoolean = true;

}else if(bTeamKillString.equals("N")){

bTeamKillBoolean = true;

}else{

bTeamKillBoolean = false;

}

}

tBalance.setText(intRSBalanceGameT + "$");

bTeamKill.setEnabled(false);

TeamKillButtonBoolean = true;

TeamKillButtonString = "True";

if(intRSBalanceGameT < 0){

intRSBalanceGameT = 0;

}

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balance1Update = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceGameT+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET buttonTeamKill = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(source == bGrenadeUse){

boolean GrenadeUseBoolean = false;

while(GrenadeUseBoolean == false){

String GrenadeUseQuestion = inputString("Did you use grenades? Y/N");

if(GrenadeUseQuestion.equals("Y")){

if(GameTGrenade1 != null){

bGrenade1.setVisible(true);

lGrenade1.setText(GameTGrenade1);

}

if(GameTGrenade2 != null){

bGrenade2.setVisible(true);

lGrenade2.setText(GameTGrenade2);

}

if(GameTGrenade3 != null){

bGrenade3.setVisible(true);

lGrenade3.setText(GameTGrenade3);

}

if(GameTGrenade4 != null){

bGrenade4.setVisible(true);

lGrenade4.setText(GameTGrenade4);

}

bFinishGrenadeUse.setVisible(true);

GrenadeUseBoolean = true;

bBombPlant.setEnabled(false);

bDropPick.setEnabled(false);

bGrenadeUse.setEnabled(false);

bSuicideReward.setEnabled(false);

bTeamKill.setEnabled(false);

GrenadeUseButtonString = "True";

}else if(GrenadeUseQuestion.equals("N")){

bDropPick.setEnabled(true);

bGrenadeUse.setEnabled(false);

GrenadeUseButtonBoolean = true;

GrenadeUseButtonString = "True";

if(BombPlantButtonString.equals("False") && buttonBombPlantString.equals("False")){

bBombPlant.setEnabled(true);

}

if(buttonSuicideRewardString.equals("False")){

bSuicideReward.setEnabled(true);

}

if(buttonTeamKillString.equals("False")){

bTeamKill.setEnabled(true);

}

GrenadeUseBoolean = true;

}else{

GrenadeUseBoolean = false;

}

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET buttonGrenadeUse = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

}

if(source == bFinishGrenadeUse){

bGrenade1.setVisible(false);

bGrenade2.setVisible(false);

bGrenade3.setVisible(false);

bGrenade4.setVisible(false);

bFinishGrenadeUse.setVisible(false);

lGrenade1.setText("");

lGrenade2.setText("");

lGrenade3.setText("");

lGrenade4.setText("");

bDropPick.setEnabled(true);

GrenadeUseButtonBoolean = true;

GrenadeUseButtonString = "True";

if(BombPlantButtonString.equals("False") && buttonBombPlantString.equals("False")){

bBombPlant.setEnabled(true);

}

if(buttonSuicideRewardString.equals("False")){

bSuicideReward.setEnabled(true);

}

if(buttonTeamKillString.equals("False")){

bTeamKill.setEnabled(true);

}

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET buttonGrenadeUse = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "True");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(source == bDropPick){

new GameTPickDrop();

dispose();

}

if(TeamKillButtonString.equals("True") && SuicideRewardButtonString.equals("True") && BombPlantButtonString.equals("True") && GrenadeUseButtonString.equals("True")){ //alias

bQuitbuyMenuGameTMenu.setEnabled(true);

}

if(source == bQuitbuyMenuGameTMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET MainGameCondition = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Result");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new MainGameT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class killMenuT extends EasyApp

{

public static void main(String[] args)

{

new killMenuT();

}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropPick = addButton("Did you pick/drop the weapon?",0,420,220,100,this);

Button bKill1 = addButton("Kill №1",366,120,220,100,this);

Button bKill2 = addButton("Kill №2",366,220,220,100,this);

Button bKill3 = addButton("Kill №3",366,320,220,100,this);

Button bKill4 = addButton("Kill №4",366,420,220,100,this);

Button bKill5 = addButton("Kill №5",366,520,220,100,this);

Button bQuitKillTMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int intRSBalanceGameT;

int intSQLRoundCTGameT;

int intSQLRoundTGameT;

int intSQLRoundNumberGameT;

int intSQLKillMenuKillNumber;

String killMenuWeapon1;

String killMenuWeapon2;

String killMenuZeus;

String killMenuGrenade1;

String killMenuGrenade2;

String killMenuGrenade3;

String killMenuGrenade4;

int killNumber;

int intSQLKillMenuKillEntryNumber;

public killMenuT() // Constructor - change window appearance

{

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCount = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCount.next() ){

String sqlRoundCountstring = sqlRoundCount.getString(1);

intSQLRoundNumberGameT = Integer.parseInt (sqlRoundCountstring);}

ResultSet rsBalance = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalance.next() ){

String rs1Balancestring = rsBalance.getString(1);

intRSBalanceGameT = Integer.parseInt (rs1Balancestring);}

ResultSet sqlRoundCT = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCT.next() ){

String sqlRoundCTstring = sqlRoundCT.getString(1);

intSQLRoundCTGameT = Integer.parseInt (sqlRoundCTstring);}

ResultSet sqlRoundT = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundT.next() ){

String sqlRoundTstring = sqlRoundT.getString(1);

intSQLRoundTGameT = Integer.parseInt (sqlRoundTstring);}

ResultSet sqlWeapon1KillMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1KillMenu .next() ){

killMenuWeapon1 = sqlWeapon1KillMenu .getString(1);

}

ResultSet sqlWeapon2KillMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2KillMenu .next() ){

killMenuWeapon2 = sqlWeapon2KillMenu .getString(1);

}

ResultSet sqlZeusKillMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeusKillMenu .next() ){

killMenuZeus = sqlZeusKillMenu .getString(1);

}

ResultSet sqlGrenade1KillMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1KillMenu .next() ){

killMenuGrenade1 = sqlGrenade1KillMenu .getString(1);

}

ResultSet sqlGrenade2KillMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2KillMenu .next() ){

killMenuGrenade2 = sqlGrenade2KillMenu .getString(1);

}

ResultSet sqlGrenade3KillMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3KillMenu .next() ){

killMenuGrenade3 = sqlGrenade3KillMenu .getString(1);

}

ResultSet sqlGrenade4KillMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4KillMenu .next() ){

killMenuGrenade4 = sqlGrenade4KillMenu .getString(1);

}

ResultSet sqlKillNumber = stmt.executeQuery( "SELECT KillNumber FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlKillNumber .next() ){

String killMenuKillNumber = sqlKillNumber.getString(1);

intSQLKillMenuKillNumber = Integer.parseInt (killMenuKillNumber);

}

ResultSet sqlKillEntryNumber = stmt.executeQuery( "SELECT KillEntryNumber FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlKillEntryNumber.next() ){

String killMenuKillEntryNumber = sqlKillEntryNumber.getString(1);

intSQLKillMenuKillEntryNumber = Integer.parseInt (killMenuKillEntryNumber);

}

//finish it

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(intSQLRoundNumberGameT + "");

tBalance.setText(intRSBalanceGameT + "$");

tCTscore.setText(intSQLRoundCTGameT + "");

tTscore.setText(intSQLRoundTGameT+ "");

bKill1.setVisible(false);

bKill2.setVisible(false);

bKill3.setVisible(false);

bKill4.setVisible(false);

bKill5.setVisible(false);

bKill1.setEnabled(false);

bKill2.setEnabled(false);

bKill3.setEnabled(false);

bKill4.setEnabled(false);

bKill5.setEnabled(false);

bQuitKillTMenu.setEnabled(false);

if(intSQLKillMenuKillNumber >= 1 && intSQLKillMenuKillEntryNumber == 0){

intSQLKillMenuKillEntryNumber = 1;

}

if(intSQLKillMenuKillNumber == 1){

bKill1.setVisible(true);

}

else if(intSQLKillMenuKillNumber == 2){

bKill1.setVisible(true);

bKill2.setVisible(true);

}

else if(intSQLKillMenuKillNumber == 3){

bKill1.setVisible(true);

bKill2.setVisible(true);

bKill3.setVisible(true);

}

else if(intSQLKillMenuKillNumber == 4){

bKill1.setVisible(true);

bKill2.setVisible(true);

bKill3.setVisible(true);

bKill4.setVisible(true);

}

else if(intSQLKillMenuKillNumber == 5){

bKill1.setVisible(true);

bKill2.setVisible(true);

bKill3.setVisible(true);

bKill4.setVisible(true);

bKill5.setVisible(true);

}

if(intSQLKillMenuKillEntryNumber > intSQLKillMenuKillNumber){

bQuitKillTMenu.setEnabled(true);

}

if(intSQLKillMenuKillEntryNumber == 1){

bKill1.setEnabled(true);

}else if(intSQLKillMenuKillEntryNumber == 2){

bKill2.setEnabled(true);

}else if(intSQLKillMenuKillEntryNumber == 3){

bKill3.setEnabled(true);

}else if(intSQLKillMenuKillEntryNumber == 4){

bKill4.setEnabled(true);

}else if(intSQLKillMenuKillEntryNumber == 5){

bKill5.setEnabled(true);

}

}

public void actions(Object source,String command){

if(source == bKill1){

intSQLKillMenuKillEntryNumber = 2;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill1KillEntryNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillEntryNumber = "+intSQLKillMenuKillEntryNumber+" WHERE idNumCS = 1;");

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new Kill1();

dispose();

}

if(source == bKill2){

intSQLKillMenuKillEntryNumber = 3;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill2KillEntryNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillEntryNumber = "+intSQLKillMenuKillEntryNumber+" WHERE idNumCS = 1;");

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new Kill2(); //change to Kill2

dispose();

}

if(source == bKill3){

intSQLKillMenuKillEntryNumber = 4;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill3KillEntryNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillEntryNumber = "+intSQLKillMenuKillEntryNumber+" WHERE idNumCS = 1;");

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new Kill3(); //change to Kill3

dispose();

}

if(source == bKill4){

intSQLKillMenuKillEntryNumber = 5;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill4KillEntryNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillEntryNumber = "+intSQLKillMenuKillEntryNumber+" WHERE idNumCS = 1;");

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new Kill4(); //change to Kill4

dispose();

}

if(source == bKill5){

intSQLKillMenuKillEntryNumber = 6;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill5KillEntryNumberUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET KillEntryNumber = "+intSQLKillMenuKillEntryNumber+" WHERE idNumCS = 1;");

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new Kill5(); //change to Kill5

dispose();

}

if(source == bDropPick){

new killMenuTPickDrop();

dispose();

}

if(source == bQuitKillTMenu){

new GameT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class Kill1 extends EasyApp

{

public static void main(String[] args)

{ new Kill1();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bKill1Weapon1 = addButton("Secondary weapon",400,90,183,50,this);

Button bKill1Weapon2 = addButton("Primary weapon",400,140,183,50,this);

Button bKill1Zeus = addButton("Zeus",400,190,183,50,this);

Button bKill1Grenade1 = addButton("Grenade 1",400,240,183,50,this);

Button bKill1Grenade2 = addButton("Grenade 2",400,290,183,50,this);

Button bKill1Grenade3 = addButton("Grenade 3",400,340,183,50,this);

Button bKill1Grenade4 = addButton("Grenade 4",400,390,183,50,this);

Button bKill1Knife = addButton("Knife",400,440,183,50,this);

Button bQuitkillMenuKill1Menu = addButton("Go Back",830,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lKill1Weapon1 = addLabel("",337,100,62,30,this);

Label lKill1Weapon2 = addLabel("",337,200,62,30,this);

Label lKill1Zeus = addLabel("",337,300,62,30,this);

Label lKill1Grenade1 = addLabel("",337,400,62,30,this);

Label lKill1Grenade2 = addLabel("",337,500,62,30,this);

Label lKill1Grenade3 = addLabel("",337,600,62,30,this);

Label lKill1Grenade4 = addLabel("",337,700,62,30,this);

int RoundCountkillMenuKill1;

int BalancekillMenuKill1;

int RoundCTkillMenuKill1;

int RoundTkillMenuKill1;

String killMenuKill1Weapon1;

String killMenuKill1Weapon2;

String killMenuKill1Zeus;

String killMenuKill1Grenade1;

String killMenuKill1Grenade2;

String killMenuKill1Grenade3;

String killMenuKill1Grenade4;

public Kill1(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuKill1Menu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuKill1Menu.next() ){

String sqlRoundCountstringkillMenuKill1Menu = sqlRoundCountkillMenuKill1Menu.getString(1);

RoundCountkillMenuKill1 = Integer.parseInt (sqlRoundCountstringkillMenuKill1Menu);}

ResultSet rsBalancekillMenuKill1Menu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuKill1Menu.next() ){

String rs1BalancestringkillMenuKill1Menu = rsBalancekillMenuKill1Menu.getString(1);

BalancekillMenuKill1 = Integer.parseInt (rs1BalancestringkillMenuKill1Menu);}

ResultSet sqlRoundCTkillMenuKill1Menu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuKill1Menu.next() ){

String sqlRoundCTstringkillMenuKill1Menu = sqlRoundCTkillMenuKill1Menu.getString(1);

RoundCTkillMenuKill1 = Integer.parseInt (sqlRoundCTstringkillMenuKill1Menu);}

ResultSet sqlRoundTkillMenuKill1Menu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuKill1Menu .next() ){

String sqlRoundTstringkillMenuKill1Menu = sqlRoundTkillMenuKill1Menu .getString(1);

RoundTkillMenuKill1 = Integer.parseInt (sqlRoundTstringkillMenuKill1Menu);}

ResultSet sqlWeapon1killMenuKill1Menu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuKill1Menu .next() ){

killMenuKill1Weapon1 = sqlWeapon1killMenuKill1Menu .getString(1);

}

ResultSet sqlWeapon2killMenuKill1Menu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuKill1Menu .next() ){

killMenuKill1Weapon2 = sqlWeapon2killMenuKill1Menu .getString(1);

}

ResultSet sqlZeuskillMenuKill1Menu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuKill1Menu .next() ){

killMenuKill1Zeus = sqlZeuskillMenuKill1Menu .getString(1);

}

ResultSet sqlGrenade1killMenuKill1Menu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuKill1Menu .next() ){

killMenuKill1Grenade1 = sqlGrenade1killMenuKill1Menu .getString(1);

}

ResultSet sqlGrenade2killMenuKill1Menu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuKill1Menu .next() ){

killMenuKill1Grenade2 = sqlGrenade2killMenuKill1Menu .getString(1);

}

ResultSet sqlGrenade3killMenuKill1Menu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuKill1Menu .next() ){

killMenuKill1Grenade3 = sqlGrenade3killMenuKill1Menu .getString(1);

}

ResultSet sqlGrenade4killMenuKill1Menu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuKill1Menu .next() ){

killMenuKill1Grenade4 = sqlGrenade4killMenuKill1Menu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuKill1 + "");

tBalance.setText(BalancekillMenuKill1 + "$");

tCTscore.setText(RoundCTkillMenuKill1 + "");

tTscore.setText(RoundTkillMenuKill1 + "");

lKill1Weapon1.setText(killMenuKill1Weapon1);

lKill1Weapon2.setText(killMenuKill1Weapon2);

lKill1Zeus.setText(killMenuKill1Zeus);

lKill1Grenade1.setText(killMenuKill1Grenade1);

lKill1Grenade2.setText(killMenuKill1Grenade2);

lKill1Grenade3.setText(killMenuKill1Grenade3);

lKill1Grenade4.setText(killMenuKill1Grenade4);

bQuitkillMenuKill1Menu.setEnabled(false);

if(killMenuKill1Weapon1 == null){

bKill1Weapon1.setEnabled(false);

}

if(killMenuKill1Weapon2 == null){

bKill1Weapon2.setEnabled(false);

}

if(killMenuKill1Zeus == null){

bKill1Zeus.setEnabled(false);

}

if(killMenuKill1Grenade1 == null){

bKill1Grenade1.setEnabled(false);

}

if(killMenuKill1Grenade2 == null){

bKill1Grenade2.setEnabled(false);

}

if(killMenuKill1Grenade3 == null){

bKill1Grenade3.setEnabled(false);

}

if(killMenuKill1Grenade4 == null){

bKill1Grenade4.setEnabled(false);

}

}

public void actions(Object source,String command){

if (source == bKill1Weapon1){

if(killMenuKill1Weapon1.equals("Glock-18")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("Dual Berettas")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("P250")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("CZ75-Auto")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 100;

}

if(killMenuKill1Weapon1.equals("Desert Eagle")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("R8 Revolver")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("Tec-9")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("USP-S")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("P2000")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon1.equals("Five-SeveN")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

bKill1Knife.setEnabled(false);

}

if (source == bKill1Weapon2){

if(killMenuKill1Weapon2.equals("MAC-10")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 600;

}

if(killMenuKill1Weapon2.equals("MP5-SD")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 600;

}

if(killMenuKill1Weapon2.equals("MP7")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 600;

}

if(killMenuKill1Weapon2.equals("UMP-45")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 600;

}

if(killMenuKill1Weapon2.equals("P90")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("PP-Bizon")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 600;

}

if(killMenuKill1Weapon2.equals("MP9")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 600;

}

if(killMenuKill1Weapon2.equals("Galil AR")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("AK-47")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("SSG 08")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("SG 553")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("AWP")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 100;

}

if(killMenuKill1Weapon2.equals("G3SG1")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("FAMAS")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("M4A4")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("M4A1-S")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("AUG")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("SCAR-20")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("Nova")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 900;

}

if(killMenuKill1Weapon2.equals("XM1014")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 900;

}

if(killMenuKill1Weapon2.equals("Sawed-Off")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 900;

}

if(killMenuKill1Weapon2.equals("M249")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("Negev")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

}

if(killMenuKill1Weapon2.equals("MAG-7")){

BalancekillMenuKill1 = BalancekillMenuKill1 + 900;

}

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

}

if (source == bKill1Zeus){

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

killMenuKill1Zeus = null;

}

if (source == bKill1Grenade1){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

killMenuKill1Grenade1 = null;

}

if (source == bKill1Grenade2){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

killMenuKill1Grenade2 = null;

}

if (source == bKill1Grenade3){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

killMenuKill1Grenade3 = null;

}

if (source == bKill1Grenade4){

BalancekillMenuKill1 = BalancekillMenuKill1 + 300;

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

killMenuKill1Grenade4 = null;

}

if (source == bKill1Knife){

BalancekillMenuKill1 = BalancekillMenuKill1 + 1500;

if(BalancekillMenuKill1 > 16000){

BalancekillMenuKill1 = 16000;

}

tBalance.setText(BalancekillMenuKill1 + "$");

bKill1Weapon1.setEnabled(false);

bKill1Weapon2.setEnabled(false);

bKill1Zeus.setEnabled(false);

bKill1Grenade1.setEnabled(false);

bKill1Grenade2.setEnabled(false);

bKill1Grenade3.setEnabled(false);

bKill1Grenade4.setEnabled(false);

bKill1Knife.setEnabled(false);

bQuitkillMenuKill1Menu.setEnabled(true);

}

if (source == bQuitkillMenuKill1Menu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill1balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+BalancekillMenuKill1+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuKill1Grenade1);

preparedStatement.setString(2,killMenuKill1Grenade2);

preparedStatement.setString(3,killMenuKill1Grenade3);

preparedStatement.setString(4,killMenuKill1Grenade4);

preparedStatement.setString(5,killMenuKill1Zeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class Kill2 extends EasyApp

{

public static void main(String[] args)

{ new Kill2();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bKill2Weapon1 = addButton("Secondary weapon",400,90,183,50,this);

Button bKill2Weapon2 = addButton("Primary weapon",400,140,183,50,this);

Button bKill2Zeus = addButton("Zeus",400,190,183,50,this);

Button bKill2Grenade1 = addButton("Grenade 1",400,240,183,50,this);

Button bKill2Grenade2 = addButton("Grenade 2",400,290,183,50,this);

Button bKill2Grenade3 = addButton("Grenade 3",400,340,183,50,this);

Button bKill2Grenade4 = addButton("Grenade 4",400,390,183,50,this);

Button bKill2Knife = addButton("Knife",400,440,183,50,this);

Button bQuitkillMenuKill2Menu = addButton("Go Back",830,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lKill2Weapon1 = addLabel("",337,100,62,30,this);

Label lKill2Weapon2 = addLabel("",337,200,62,30,this);

Label lKill2Zeus = addLabel("",337,300,62,30,this);

Label lKill2Grenade1 = addLabel("",337,400,62,30,this);

Label lKill2Grenade2 = addLabel("",337,500,62,30,this);

Label lKill2Grenade3 = addLabel("",337,600,62,30,this);

Label lKill2Grenade4 = addLabel("",337,700,62,30,this);

int RoundCountkillMenuKill2;

int BalancekillMenuKill2;

int RoundCTkillMenuKill2;

int RoundTkillMenuKill2;

String killMenuKill2Weapon1;

String killMenuKill2Weapon2;

String killMenuKill2Zeus;

String killMenuKill2Grenade1;

String killMenuKill2Grenade2;

String killMenuKill2Grenade3;

String killMenuKill2Grenade4;

public Kill2(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuKill2Menu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuKill2Menu.next() ){

String sqlRoundCountstringkillMenuKill2Menu = sqlRoundCountkillMenuKill2Menu.getString(1);

RoundCountkillMenuKill2 = Integer.parseInt (sqlRoundCountstringkillMenuKill2Menu);}

ResultSet rsBalancekillMenuKill2Menu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuKill2Menu.next() ){

String rs1BalancestringkillMenuKill2Menu = rsBalancekillMenuKill2Menu.getString(1);

BalancekillMenuKill2 = Integer.parseInt (rs1BalancestringkillMenuKill2Menu);}

ResultSet sqlRoundCTkillMenuKill2Menu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuKill2Menu.next() ){

String sqlRoundCTstringkillMenuKill2Menu = sqlRoundCTkillMenuKill2Menu.getString(1);

RoundCTkillMenuKill2 = Integer.parseInt (sqlRoundCTstringkillMenuKill2Menu);}

ResultSet sqlRoundTkillMenuKill2Menu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuKill2Menu .next() ){

String sqlRoundTstringkillMenuKill2Menu = sqlRoundTkillMenuKill2Menu .getString(1);

RoundTkillMenuKill2 = Integer.parseInt (sqlRoundTstringkillMenuKill2Menu);}

ResultSet sqlWeapon1killMenuKill2Menu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuKill2Menu .next() ){

killMenuKill2Weapon1 = sqlWeapon1killMenuKill2Menu .getString(1);

}

ResultSet sqlWeapon2killMenuKill2Menu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuKill2Menu .next() ){

killMenuKill2Weapon2 = sqlWeapon2killMenuKill2Menu .getString(1);

}

ResultSet sqlZeuskillMenuKill2Menu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuKill2Menu .next() ){

killMenuKill2Zeus = sqlZeuskillMenuKill2Menu .getString(1);

}

ResultSet sqlGrenade1killMenuKill2Menu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuKill2Menu .next() ){

killMenuKill2Grenade1 = sqlGrenade1killMenuKill2Menu .getString(1);

}

ResultSet sqlGrenade2killMenuKill2Menu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuKill2Menu .next() ){

killMenuKill2Grenade2 = sqlGrenade2killMenuKill2Menu .getString(1);

}

ResultSet sqlGrenade3killMenuKill2Menu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuKill2Menu .next() ){

killMenuKill2Grenade3 = sqlGrenade3killMenuKill2Menu .getString(1);

}

ResultSet sqlGrenade4killMenuKill2Menu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuKill2Menu .next() ){

killMenuKill2Grenade4 = sqlGrenade4killMenuKill2Menu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuKill2 + "");

tBalance.setText(BalancekillMenuKill2 + "$");

tCTscore.setText(RoundCTkillMenuKill2 + "");

tTscore.setText(RoundTkillMenuKill2 + "");

lKill2Weapon1.setText(killMenuKill2Weapon1);

lKill2Weapon2.setText(killMenuKill2Weapon2);

lKill2Zeus.setText(killMenuKill2Zeus);

lKill2Grenade1.setText(killMenuKill2Grenade1);

lKill2Grenade2.setText(killMenuKill2Grenade2);

lKill2Grenade3.setText(killMenuKill2Grenade3);

lKill2Grenade4.setText(killMenuKill2Grenade4);

bQuitkillMenuKill2Menu.setEnabled(false);

if(killMenuKill2Weapon1 == null){

bKill2Weapon1.setEnabled(false);

}

if(killMenuKill2Weapon2 == null){

bKill2Weapon2.setEnabled(false);

}

if(killMenuKill2Zeus == null){

bKill2Zeus.setEnabled(false);

}

if(killMenuKill2Grenade1 == null){

bKill2Grenade1.setEnabled(false);

}

if(killMenuKill2Grenade2 == null){

bKill2Grenade2.setEnabled(false);

}

if(killMenuKill2Grenade3 == null){

bKill2Grenade3.setEnabled(false);

}

if(killMenuKill2Grenade4 == null){

bKill2Grenade4.setEnabled(false);

}

}

public void actions(Object source,String command){

if (source == bKill2Weapon1){

if(killMenuKill2Weapon1.equals("Glock-18")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("Dual Berettas")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("P250")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("CZ75-Auto")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 100;

}

if(killMenuKill2Weapon1.equals("Desert Eagle")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("R8 Revolver")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("Tec-9")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("USP-S")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("P2000")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon1.equals("Five-SeveN")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

}

if (source == bKill2Weapon2){

if(killMenuKill2Weapon2.equals("MAC-10")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 600;

}

if(killMenuKill2Weapon2.equals("MP5-SD")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 600;

}

if(killMenuKill2Weapon2.equals("MP7")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 600;

}

if(killMenuKill2Weapon2.equals("UMP-45")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 600;

}

if(killMenuKill2Weapon2.equals("P90")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("PP-Bizon")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 600;

}

if(killMenuKill2Weapon2.equals("MP9")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 600;

}

if(killMenuKill2Weapon2.equals("Galil AR")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("AK-47")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("SSG 08")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("SG 553")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("AWP")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 100;

}

if(killMenuKill2Weapon2.equals("G3SG1")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("FAMAS")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("M4A4")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("M4A1-S")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("AUG")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("SCAR-20")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("Nova")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 900;

}

if(killMenuKill2Weapon2.equals("XM1014")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 900;

}

if(killMenuKill2Weapon2.equals("Sawed-Off")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 900;

}

if(killMenuKill2Weapon2.equals("M249")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("Negev")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

}

if(killMenuKill2Weapon2.equals("MAG-7")){

BalancekillMenuKill2 = BalancekillMenuKill2 + 900;

}

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

}

if (source == bKill2Zeus){

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

killMenuKill2Zeus = null;

}

if (source == bKill2Grenade1){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

killMenuKill2Grenade1 = null;

}

if (source == bKill2Grenade2){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

killMenuKill2Grenade2 = null;

}

if (source == bKill2Grenade3){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

killMenuKill2Grenade3 = null;

}

if (source == bKill2Grenade4){

BalancekillMenuKill2 = BalancekillMenuKill2 + 300;

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

killMenuKill2Grenade4 = null;

}

if (source == bKill2Knife){

BalancekillMenuKill2 = BalancekillMenuKill2 + 1500;

if(BalancekillMenuKill2 > 16000){

BalancekillMenuKill2 = 16000;

}

tBalance.setText(BalancekillMenuKill2 + "$");

bKill2Weapon1.setEnabled(false);

bKill2Weapon2.setEnabled(false);

bKill2Zeus.setEnabled(false);

bKill2Grenade1.setEnabled(false);

bKill2Grenade2.setEnabled(false);

bKill2Grenade3.setEnabled(false);

bKill2Grenade4.setEnabled(false);

bKill2Knife.setEnabled(false);

bQuitkillMenuKill2Menu.setEnabled(true);

}

if (source == bQuitkillMenuKill2Menu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill1balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+BalancekillMenuKill2+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuKill2Grenade1);

preparedStatement.setString(2,killMenuKill2Grenade2);

preparedStatement.setString(3,killMenuKill2Grenade3);

preparedStatement.setString(4,killMenuKill2Grenade4);

preparedStatement.setString(5,killMenuKill2Zeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class Kill3 extends EasyApp

{

public static void main(String[] args)

{ new Kill3();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bKill3Weapon1 = addButton("Secondary weapon",400,90,183,50,this);

Button bKill3Weapon2 = addButton("Primary weapon",400,140,183,50,this);

Button bKill3Zeus = addButton("Zeus",400,190,183,50,this);

Button bKill3Grenade1 = addButton("Grenade 1",400,240,183,50,this);

Button bKill3Grenade2 = addButton("Grenade 2",400,290,183,50,this);

Button bKill3Grenade3 = addButton("Grenade 3",400,340,183,50,this);

Button bKill3Grenade4 = addButton("Grenade 4",400,390,183,50,this);

Button bKill3Knife = addButton("Knife",400,440,183,50,this);

Button bQuitkillMenuKill3Menu = addButton("Go Back",830,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lKill3Weapon1 = addLabel("",337,100,62,30,this);

Label lKill3Weapon2 = addLabel("",337,200,62,30,this);

Label lKill3Zeus = addLabel("",337,300,62,30,this);

Label lKill3Grenade1 = addLabel("",337,400,62,30,this);

Label lKill3Grenade2 = addLabel("",337,500,62,30,this);

Label lKill3Grenade3 = addLabel("",337,600,62,30,this);

Label lKill3Grenade4 = addLabel("",337,700,62,30,this);

int RoundCountkillMenuKill3;

int BalancekillMenuKill3;

int RoundCTkillMenuKill3;

int RoundTkillMenuKill3;

String killMenuKill3Weapon1;

String killMenuKill3Weapon2;

String killMenuKill3Zeus;

String killMenuKill3Grenade1;

String killMenuKill3Grenade2;

String killMenuKill3Grenade3;

String killMenuKill3Grenade4;

public Kill3(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuKill3Menu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuKill3Menu.next() ){

String sqlRoundCountstringkillMenuKill3Menu = sqlRoundCountkillMenuKill3Menu.getString(1);

RoundCountkillMenuKill3 = Integer.parseInt (sqlRoundCountstringkillMenuKill3Menu);}

ResultSet rsBalancekillMenuKill3Menu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuKill3Menu.next() ){

String rs1BalancestringkillMenuKill3Menu = rsBalancekillMenuKill3Menu.getString(1);

BalancekillMenuKill3 = Integer.parseInt (rs1BalancestringkillMenuKill3Menu);}

ResultSet sqlRoundCTkillMenuKill3Menu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuKill3Menu.next() ){

String sqlRoundCTstringkillMenuKill3Menu = sqlRoundCTkillMenuKill3Menu.getString(1);

RoundCTkillMenuKill3 = Integer.parseInt (sqlRoundCTstringkillMenuKill3Menu);}

ResultSet sqlRoundTkillMenuKill3Menu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuKill3Menu .next() ){

String sqlRoundTstringkillMenuKill3Menu = sqlRoundTkillMenuKill3Menu .getString(1);

RoundTkillMenuKill3 = Integer.parseInt (sqlRoundTstringkillMenuKill3Menu);}

ResultSet sqlWeapon1killMenuKill3Menu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuKill3Menu .next() ){

killMenuKill3Weapon1 = sqlWeapon1killMenuKill3Menu .getString(1);

}

ResultSet sqlWeapon2killMenuKill3Menu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuKill3Menu .next() ){

killMenuKill3Weapon2 = sqlWeapon2killMenuKill3Menu .getString(1);

}

ResultSet sqlZeuskillMenuKill3Menu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuKill3Menu .next() ){

killMenuKill3Zeus = sqlZeuskillMenuKill3Menu .getString(1);

}

ResultSet sqlGrenade1killMenuKill3Menu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuKill3Menu .next() ){

killMenuKill3Grenade1 = sqlGrenade1killMenuKill3Menu .getString(1);

}

ResultSet sqlGrenade2killMenuKill3Menu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuKill3Menu .next() ){

killMenuKill3Grenade2 = sqlGrenade2killMenuKill3Menu .getString(1);

}

ResultSet sqlGrenade3killMenuKill3Menu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuKill3Menu .next() ){

killMenuKill3Grenade3 = sqlGrenade3killMenuKill3Menu .getString(1);

}

ResultSet sqlGrenade4killMenuKill3Menu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuKill3Menu .next() ){

killMenuKill3Grenade4 = sqlGrenade4killMenuKill3Menu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuKill3 + "");

tBalance.setText(BalancekillMenuKill3 + "$");

tCTscore.setText(RoundCTkillMenuKill3 + "");

tTscore.setText(RoundTkillMenuKill3 + "");

lKill3Weapon1.setText(killMenuKill3Weapon1);

lKill3Weapon2.setText(killMenuKill3Weapon2);

lKill3Zeus.setText(killMenuKill3Zeus);

lKill3Grenade1.setText(killMenuKill3Grenade1);

lKill3Grenade2.setText(killMenuKill3Grenade2);

lKill3Grenade3.setText(killMenuKill3Grenade3);

lKill3Grenade4.setText(killMenuKill3Grenade4);

bQuitkillMenuKill3Menu.setEnabled(false);

if(killMenuKill3Weapon1 == null){

bKill3Weapon1.setEnabled(false);

}

if(killMenuKill3Weapon2 == null){

bKill3Weapon2.setEnabled(false);

}

if(killMenuKill3Zeus == null){

bKill3Zeus.setEnabled(false);

}

if(killMenuKill3Grenade1 == null){

bKill3Grenade1.setEnabled(false);

}

if(killMenuKill3Grenade2 == null){

bKill3Grenade2.setEnabled(false);

}

if(killMenuKill3Grenade3 == null){

bKill3Grenade3.setEnabled(false);

}

if(killMenuKill3Grenade4 == null){

bKill3Grenade4.setEnabled(false);

}

}

public void actions(Object source,String command){

if (source == bKill3Weapon1){

if(killMenuKill3Weapon1.equals("Glock-18")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("Dual Berettas")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("P250")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("CZ75-Auto")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 100;

}

if(killMenuKill3Weapon1.equals("Desert Eagle")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("R8 Revolver")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("Tec-9")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("USP-S")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("P2000")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon1.equals("Five-SeveN")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

}

if (source == bKill3Weapon2){

if(killMenuKill3Weapon2.equals("MAC-10")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 600;

}

if(killMenuKill3Weapon2.equals("MP5-SD")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 600;

}

if(killMenuKill3Weapon2.equals("MP7")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 600;

}

if(killMenuKill3Weapon2.equals("UMP-45")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 600;

}

if(killMenuKill3Weapon2.equals("P90")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("PP-Bizon")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 600;

}

if(killMenuKill3Weapon2.equals("MP9")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 600;

}

if(killMenuKill3Weapon2.equals("Galil AR")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("AK-47")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("SSG 08")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("SG 553")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("AWP")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 100;

}

if(killMenuKill3Weapon2.equals("G3SG1")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("FAMAS")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("M4A4")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("M4A1-S")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("AUG")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("SCAR-20")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("Nova")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 900;

}

if(killMenuKill3Weapon2.equals("XM1014")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 900;

}

if(killMenuKill3Weapon2.equals("Sawed-Off")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 900;

}

if(killMenuKill3Weapon2.equals("M249")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("Negev")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

}

if(killMenuKill3Weapon2.equals("MAG-7")){

BalancekillMenuKill3 = BalancekillMenuKill3 + 900;

}

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

}

if (source == bKill3Zeus){

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

killMenuKill3Zeus = null;

}

if (source == bKill3Grenade1){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

killMenuKill3Grenade1 = null;

}

if (source == bKill3Grenade2){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

killMenuKill3Grenade2 = null;

}

if (source == bKill3Grenade3){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

killMenuKill3Grenade3 = null;

}

if (source == bKill3Grenade4){

BalancekillMenuKill3 = BalancekillMenuKill3 + 300;

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

killMenuKill3Grenade4 = null;

}

if (source == bKill3Knife){

BalancekillMenuKill3 = BalancekillMenuKill3 + 1500;

if(BalancekillMenuKill3 > 16000){

BalancekillMenuKill3 = 16000;

}

tBalance.setText(BalancekillMenuKill3 + "$");

bKill3Weapon1.setEnabled(false);

bKill3Weapon2.setEnabled(false);

bKill3Zeus.setEnabled(false);

bKill3Grenade1.setEnabled(false);

bKill3Grenade2.setEnabled(false);

bKill3Grenade3.setEnabled(false);

bKill3Grenade4.setEnabled(false);

bKill3Knife.setEnabled(false);

bQuitkillMenuKill3Menu.setEnabled(true);

}

if (source == bQuitkillMenuKill3Menu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill1balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+BalancekillMenuKill3+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuKill3Grenade1);

preparedStatement.setString(2,killMenuKill3Grenade2);

preparedStatement.setString(3,killMenuKill3Grenade3);

preparedStatement.setString(4,killMenuKill3Grenade4);

preparedStatement.setString(5,killMenuKill3Zeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class Kill4 extends EasyApp

{

public static void main(String[] args)

{ new Kill4();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bKill4Weapon1 = addButton("Secondary weapon",400,90,183,50,this);

Button bKill4Weapon2 = addButton("Primary weapon",400,140,183,50,this);

Button bKill4Zeus = addButton("Zeus",400,190,183,50,this);

Button bKill4Grenade1 = addButton("Grenade 1",400,240,183,50,this);

Button bKill4Grenade2 = addButton("Grenade 2",400,290,183,50,this);

Button bKill4Grenade3 = addButton("Grenade 3",400,340,183,50,this);

Button bKill4Grenade4 = addButton("Grenade 4",400,390,183,50,this);

Button bKill4Knife = addButton("Knife",400,440,183,50,this);

Button bQuitkillMenuKill4Menu = addButton("Go Back",830,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lKill4Weapon1 = addLabel("",337,100,62,30,this);

Label lKill4Weapon2 = addLabel("",337,200,62,30,this);

Label lKill4Zeus = addLabel("",337,300,62,30,this);

Label lKill4Grenade1 = addLabel("",337,400,62,30,this);

Label lKill4Grenade2 = addLabel("",337,500,62,30,this);

Label lKill4Grenade3 = addLabel("",337,600,62,30,this);

Label lKill4Grenade4 = addLabel("",337,700,62,30,this);

int RoundCountkillMenuKill4;

int BalancekillMenuKill4;

int RoundCTkillMenuKill4;

int RoundTkillMenuKill4;

String killMenuKill4Weapon1;

String killMenuKill4Weapon2;

String killMenuKill4Zeus;

String killMenuKill4Grenade1;

String killMenuKill4Grenade2;

String killMenuKill4Grenade3;

String killMenuKill4Grenade4;

public Kill4(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuKill4Menu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuKill4Menu.next() ){

String sqlRoundCountstringkillMenuKill4Menu = sqlRoundCountkillMenuKill4Menu.getString(1);

RoundCountkillMenuKill4 = Integer.parseInt (sqlRoundCountstringkillMenuKill4Menu);}

ResultSet rsBalancekillMenuKill4Menu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuKill4Menu.next() ){

String rs1BalancestringkillMenuKill4Menu = rsBalancekillMenuKill4Menu.getString(1);

BalancekillMenuKill4 = Integer.parseInt (rs1BalancestringkillMenuKill4Menu);}

ResultSet sqlRoundCTkillMenuKill4Menu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuKill4Menu.next() ){

String sqlRoundCTstringkillMenuKill4Menu = sqlRoundCTkillMenuKill4Menu.getString(1);

RoundCTkillMenuKill4 = Integer.parseInt (sqlRoundCTstringkillMenuKill4Menu);}

ResultSet sqlRoundTkillMenuKill4Menu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuKill4Menu .next() ){

String sqlRoundTstringkillMenuKill4Menu = sqlRoundTkillMenuKill4Menu .getString(1);

RoundTkillMenuKill4 = Integer.parseInt (sqlRoundTstringkillMenuKill4Menu);}

ResultSet sqlWeapon1killMenuKill4Menu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuKill4Menu .next() ){

killMenuKill4Weapon1 = sqlWeapon1killMenuKill4Menu .getString(1);

}

ResultSet sqlWeapon2killMenuKill4Menu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuKill4Menu .next() ){

killMenuKill4Weapon2 = sqlWeapon2killMenuKill4Menu .getString(1);

}

ResultSet sqlZeuskillMenuKill4Menu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuKill4Menu .next() ){

killMenuKill4Zeus = sqlZeuskillMenuKill4Menu .getString(1);

}

ResultSet sqlGrenade1killMenuKill4Menu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuKill4Menu .next() ){

killMenuKill4Grenade1 = sqlGrenade1killMenuKill4Menu .getString(1);

}

ResultSet sqlGrenade2killMenuKill4Menu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuKill4Menu .next() ){

killMenuKill4Grenade2 = sqlGrenade2killMenuKill4Menu .getString(1);

}

ResultSet sqlGrenade3killMenuKill4Menu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuKill4Menu .next() ){

killMenuKill4Grenade3 = sqlGrenade3killMenuKill4Menu .getString(1);

}

ResultSet sqlGrenade4killMenuKill4Menu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuKill4Menu .next() ){

killMenuKill4Grenade4 = sqlGrenade4killMenuKill4Menu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuKill4 + "");

tBalance.setText(BalancekillMenuKill4 + "$");

tCTscore.setText(RoundCTkillMenuKill4 + "");

tTscore.setText(RoundTkillMenuKill4 + "");

lKill4Weapon1.setText(killMenuKill4Weapon1);

lKill4Weapon2.setText(killMenuKill4Weapon2);

lKill4Zeus.setText(killMenuKill4Zeus);

lKill4Grenade1.setText(killMenuKill4Grenade1);

lKill4Grenade2.setText(killMenuKill4Grenade2);

lKill4Grenade3.setText(killMenuKill4Grenade3);

lKill4Grenade4.setText(killMenuKill4Grenade4);

bQuitkillMenuKill4Menu.setEnabled(false);

if(killMenuKill4Weapon1 == null){

bKill4Weapon1.setEnabled(false);

}

if(killMenuKill4Weapon2 == null){

bKill4Weapon2.setEnabled(false);

}

if(killMenuKill4Zeus == null){

bKill4Zeus.setEnabled(false);

}

if(killMenuKill4Grenade1 == null){

bKill4Grenade1.setEnabled(false);

}

if(killMenuKill4Grenade2 == null){

bKill4Grenade2.setEnabled(false);

}

if(killMenuKill4Grenade3 == null){

bKill4Grenade3.setEnabled(false);

}

if(killMenuKill4Grenade4 == null){

bKill4Grenade4.setEnabled(false);

}

}

public void actions(Object source,String command){

if (source == bKill4Weapon1){

if(killMenuKill4Weapon1.equals("Glock-18")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("Dual Berettas")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("P250")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("CZ75-Auto")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 100;

}

if(killMenuKill4Weapon1.equals("Desert Eagle")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("R8 Revolver")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("Tec-9")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("USP-S")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("P2000")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon1.equals("Five-SeveN")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

}

if (source == bKill4Weapon2){

if(killMenuKill4Weapon2.equals("MAC-10")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 600;

}

if(killMenuKill4Weapon2.equals("MP5-SD")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 600;

}

if(killMenuKill4Weapon2.equals("MP7")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 600;

}

if(killMenuKill4Weapon2.equals("UMP-45")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 600;

}

if(killMenuKill4Weapon2.equals("P90")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("PP-Bizon")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 600;

}

if(killMenuKill4Weapon2.equals("MP9")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 600;

}

if(killMenuKill4Weapon2.equals("Galil AR")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("AK-47")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("SSG 08")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("SG 553")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("AWP")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 100;

}

if(killMenuKill4Weapon2.equals("G3SG1")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("FAMAS")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("M4A4")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("M4A1-S")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("AUG")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("SCAR-20")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("Nova")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 900;

}

if(killMenuKill4Weapon2.equals("XM1014")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 900;

}

if(killMenuKill4Weapon2.equals("Sawed-Off")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 900;

}

if(killMenuKill4Weapon2.equals("M249")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("Negev")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

}

if(killMenuKill4Weapon2.equals("MAG-7")){

BalancekillMenuKill4 = BalancekillMenuKill4 + 900;

}

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

}

if (source == bKill4Zeus){

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

killMenuKill4Zeus = null;

}

if (source == bKill4Grenade1){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

killMenuKill4Grenade1 = null;

}

if (source == bKill4Grenade2){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

killMenuKill4Grenade2 = null;

}

if (source == bKill4Grenade3){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

killMenuKill4Grenade3 = null;

}

if (source == bKill4Grenade4){

BalancekillMenuKill4 = BalancekillMenuKill4 + 300;

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

killMenuKill4Grenade4 = null;

}

if (source == bKill4Knife){

BalancekillMenuKill4 = BalancekillMenuKill4 + 1500;

if(BalancekillMenuKill4 > 16000){

BalancekillMenuKill4 = 16000;

}

tBalance.setText(BalancekillMenuKill4 + "$");

bKill4Weapon1.setEnabled(false);

bKill4Weapon2.setEnabled(false);

bKill4Zeus.setEnabled(false);

bKill4Grenade1.setEnabled(false);

bKill4Grenade2.setEnabled(false);

bKill4Grenade3.setEnabled(false);

bKill4Grenade4.setEnabled(false);

bKill4Knife.setEnabled(false);

bQuitkillMenuKill4Menu.setEnabled(true);

}

if (source == bQuitkillMenuKill4Menu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill1balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+BalancekillMenuKill4+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuKill4Grenade1);

preparedStatement.setString(2,killMenuKill4Grenade2);

preparedStatement.setString(3,killMenuKill4Grenade3);

preparedStatement.setString(4,killMenuKill4Grenade4);

preparedStatement.setString(5,killMenuKill4Zeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class Kill5 extends EasyApp

{

public static void main(String[] args)

{ new Kill5();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bKill5Weapon1 = addButton("Secondary weapon",400,90,183,50,this);

Button bKill5Weapon2 = addButton("Primary weapon",400,140,183,50,this);

Button bKill5Zeus = addButton("Zeus",400,190,183,50,this);

Button bKill5Grenade1 = addButton("Grenade 1",400,240,183,50,this);

Button bKill5Grenade2 = addButton("Grenade 2",400,290,183,50,this);

Button bKill5Grenade3 = addButton("Grenade 3",400,340,183,50,this);

Button bKill5Grenade4 = addButton("Grenade 4",400,390,183,50,this);

Button bKill5Knife = addButton("Knife",400,440,183,50,this);

Button bQuitkillMenuKill5Menu = addButton("Go Back",830,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lKill5Weapon1 = addLabel("",337,100,62,30,this);

Label lKill5Weapon2 = addLabel("",337,200,62,30,this);

Label lKill5Zeus = addLabel("",337,300,62,30,this);

Label lKill5Grenade1 = addLabel("",337,400,62,30,this);

Label lKill5Grenade2 = addLabel("",337,500,62,30,this);

Label lKill5Grenade3 = addLabel("",337,600,62,30,this);

Label lKill5Grenade4 = addLabel("",337,700,62,30,this);

int RoundCountkillMenuKill5;

int BalancekillMenuKill5;

int RoundCTkillMenuKill5;

int RoundTkillMenuKill5;

String killMenuKill5Weapon1;

String killMenuKill5Weapon2;

String killMenuKill5Zeus;

String killMenuKill5Grenade1;

String killMenuKill5Grenade2;

String killMenuKill5Grenade3;

String killMenuKill5Grenade4;

public Kill5(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuKill5Menu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuKill5Menu.next() ){

String sqlRoundCountstringkillMenuKill5Menu = sqlRoundCountkillMenuKill5Menu.getString(1);

RoundCountkillMenuKill5 = Integer.parseInt (sqlRoundCountstringkillMenuKill5Menu);}

ResultSet rsBalancekillMenuKill5Menu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuKill5Menu.next() ){

String rs1BalancestringkillMenuKill5Menu = rsBalancekillMenuKill5Menu.getString(1);

BalancekillMenuKill5 = Integer.parseInt (rs1BalancestringkillMenuKill5Menu);}

ResultSet sqlRoundCTkillMenuKill5Menu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuKill5Menu.next() ){

String sqlRoundCTstringkillMenuKill5Menu = sqlRoundCTkillMenuKill5Menu.getString(1);

RoundCTkillMenuKill5 = Integer.parseInt (sqlRoundCTstringkillMenuKill5Menu);}

ResultSet sqlRoundTkillMenuKill5Menu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuKill5Menu .next() ){

String sqlRoundTstringkillMenuKill5Menu = sqlRoundTkillMenuKill5Menu .getString(1);

RoundTkillMenuKill5 = Integer.parseInt (sqlRoundTstringkillMenuKill5Menu);}

ResultSet sqlWeapon1killMenuKill5Menu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuKill5Menu .next() ){

killMenuKill5Weapon1 = sqlWeapon1killMenuKill5Menu .getString(1);

}

ResultSet sqlWeapon2killMenuKill5Menu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuKill5Menu .next() ){

killMenuKill5Weapon2 = sqlWeapon2killMenuKill5Menu .getString(1);

}

ResultSet sqlZeuskillMenuKill5Menu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuKill5Menu .next() ){

killMenuKill5Zeus = sqlZeuskillMenuKill5Menu .getString(1);

}

ResultSet sqlGrenade1killMenuKill5Menu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuKill5Menu .next() ){

killMenuKill5Grenade1 = sqlGrenade1killMenuKill5Menu .getString(1);

}

ResultSet sqlGrenade2killMenuKill5Menu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuKill5Menu .next() ){

killMenuKill5Grenade2 = sqlGrenade2killMenuKill5Menu .getString(1);

}

ResultSet sqlGrenade3killMenuKill5Menu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuKill5Menu .next() ){

killMenuKill5Grenade3 = sqlGrenade3killMenuKill5Menu .getString(1);

}

ResultSet sqlGrenade4killMenuKill5Menu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuKill5Menu .next() ){

killMenuKill5Grenade4 = sqlGrenade4killMenuKill5Menu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuKill5 + "");

tBalance.setText(BalancekillMenuKill5 + "$");

tCTscore.setText(RoundCTkillMenuKill5 + "");

tTscore.setText(RoundTkillMenuKill5 + "");

lKill5Weapon1.setText(killMenuKill5Weapon1);

lKill5Weapon2.setText(killMenuKill5Weapon2);

lKill5Zeus.setText(killMenuKill5Zeus);

lKill5Grenade1.setText(killMenuKill5Grenade1);

lKill5Grenade2.setText(killMenuKill5Grenade2);

lKill5Grenade3.setText(killMenuKill5Grenade3);

lKill5Grenade4.setText(killMenuKill5Grenade4);

bQuitkillMenuKill5Menu.setEnabled(false);

if(killMenuKill5Weapon1 == null){

bKill5Weapon1.setEnabled(false);

}

if(killMenuKill5Weapon2 == null){

bKill5Weapon2.setEnabled(false);

}

if(killMenuKill5Zeus == null){

bKill5Zeus.setEnabled(false);

}

if(killMenuKill5Grenade1 == null){

bKill5Grenade1.setEnabled(false);

}

if(killMenuKill5Grenade2 == null){

bKill5Grenade2.setEnabled(false);

}

if(killMenuKill5Grenade3 == null){

bKill5Grenade3.setEnabled(false);

}

if(killMenuKill5Grenade4 == null){

bKill5Grenade4.setEnabled(false);

}

}

public void actions(Object source,String command){

if (source == bKill5Weapon1){

if(killMenuKill5Weapon1.equals("Glock-18")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("Dual Berettas")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("P250")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("CZ75-Auto")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 100;

}

if(killMenuKill5Weapon1.equals("Desert Eagle")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("R8 Revolver")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("Tec-9")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("USP-S")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("P2000")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon1.equals("Five-SeveN")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

}

if (source == bKill5Weapon2){

if(killMenuKill5Weapon2.equals("MAC-10")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 600;

}

if(killMenuKill5Weapon2.equals("MP5-SD")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 600;

}

if(killMenuKill5Weapon2.equals("MP7")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 600;

}

if(killMenuKill5Weapon2.equals("UMP-45")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 600;

}

if(killMenuKill5Weapon2.equals("P90")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("PP-Bizon")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 600;

}

if(killMenuKill5Weapon2.equals("MP9")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 600;

}

if(killMenuKill5Weapon2.equals("Galil AR")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("AK-47")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("SSG 08")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("SG 553")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("AWP")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 100;

}

if(killMenuKill5Weapon2.equals("G3SG1")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("FAMAS")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("M4A4")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("M4A1-S")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("AUG")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("SCAR-20")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("Nova")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 900;

}

if(killMenuKill5Weapon2.equals("XM1014")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 900;

}

if(killMenuKill5Weapon2.equals("Sawed-Off")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 900;

}

if(killMenuKill5Weapon2.equals("M249")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("Negev")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

}

if(killMenuKill5Weapon2.equals("MAG-7")){

BalancekillMenuKill5 = BalancekillMenuKill5 + 900;

}

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

}

if (source == bKill5Zeus){

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

killMenuKill5Zeus = null;

}

if (source == bKill5Grenade1){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

killMenuKill5Grenade1 = null;

}

if (source == bKill5Grenade2){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

killMenuKill5Grenade2 = null;

}

if (source == bKill5Grenade3){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

killMenuKill5Grenade3 = null;

}

if (source == bKill5Grenade4){

BalancekillMenuKill5 = BalancekillMenuKill5 + 300;

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

killMenuKill5Grenade4 = null;

}

if (source == bKill5Knife){

BalancekillMenuKill5 = BalancekillMenuKill5 + 1500;

if(BalancekillMenuKill5 > 16000){

BalancekillMenuKill5 = 16000;

}

tBalance.setText(BalancekillMenuKill5 + "$");

bKill5Weapon1.setEnabled(false);

bKill5Weapon2.setEnabled(false);

bKill5Zeus.setEnabled(false);

bKill5Grenade1.setEnabled(false);

bKill5Grenade2.setEnabled(false);

bKill5Grenade3.setEnabled(false);

bKill5Grenade4.setEnabled(false);

bKill5Knife.setEnabled(false);

bQuitkillMenuKill5Menu.setEnabled(true);

}

if (source == bQuitkillMenuKill5Menu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int Kill1balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+BalancekillMenuKill5+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuKill5Grenade1);

preparedStatement.setString(2,killMenuKill5Grenade2);

preparedStatement.setString(3,killMenuKill5Grenade3);

preparedStatement.setString(4,killMenuKill5Grenade4);

preparedStatement.setString(5,killMenuKill5Zeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuT();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class GameTPickDrop extends EasyApp

{

public static void main(String[] args)

{ new GameTPickDrop();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropChoice = addButton("Did you drop a weapon?",100,300,798,100,this);

Button bPickChoice = addButton("Did you pick a weapon?",100,400,798,100,this);

Button bQuitBuyMenu = addButton("Go Back",850,600,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountbuyMenuPickDrop;

int BalancebuyMenuPickDrop;

int RoundCTbuyMenuPickDrop;

int RoundTbuyMenuPickDrop;

public GameTPickDrop(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountbuyMenuPickDrop = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountbuyMenuPickDrop.next() ){

String sqlRoundCountstringbuyMenuPickDrop = sqlRoundCountbuyMenuPickDrop.getString(1);

RoundCountbuyMenuPickDrop = Integer.parseInt (sqlRoundCountstringbuyMenuPickDrop);}

ResultSet rsBalancebuyMenuPickDrop = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancebuyMenuPickDrop.next() ){

String rs1BalancestringbuyMenuPickDrop = rsBalancebuyMenuPickDrop.getString(1);

BalancebuyMenuPickDrop = Integer.parseInt (rs1BalancestringbuyMenuPickDrop);}

ResultSet sqlRoundCTbuyMenuPickDrop = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTbuyMenuPickDrop.next() ){

String sqlRoundCTstringbuyMenuPickDrop = sqlRoundCTbuyMenuPickDrop.getString(1);

RoundCTbuyMenuPickDrop = Integer.parseInt (sqlRoundCTstringbuyMenuPickDrop);}

ResultSet sqlRoundTbuyMenuPickDrop = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTbuyMenuPickDrop .next() ){

String sqlRoundTstringbuyMenuPickDrop = sqlRoundTbuyMenuPickDrop .getString(1);

RoundTbuyMenuPickDrop = Integer.parseInt (sqlRoundTstringbuyMenuPickDrop);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,700);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountbuyMenuPickDrop + "");

tBalance.setText(BalancebuyMenuPickDrop + "$");

tCTscore.setText(RoundCTbuyMenuPickDrop + "");

tTscore.setText(RoundTbuyMenuPickDrop + "");

}

public void actions(Object source,String command){

if (source == bQuitBuyMenu)

{

new GameT();

dispose();

}

if (source == bDropChoice)

{

new GameTDropMenu();

dispose();

}

if (source == bPickChoice)

{

new GameTPickMenu();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class GameTDropMenu extends EasyApp

{

public static void main(String[] args)

{ new GameTDropMenu();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,800,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropWeapon1 = addButton("Secondary weapon",400,100,183,100,this);

Button bDropWeapon2 = addButton("Primary weapon",400,200,183,100,this);

Button bDropZeus = addButton("Zeus",400,300,183,100,this);

Button bDropGrenade1 = addButton("Grenade 1",400,400,183,100,this);

Button bDropGrenade2 = addButton("Grenade 2",400,500,183,100,this);

Button bDropGrenade3 = addButton("Grenade 3",400,600,183,100,this);

Button bDropGrenade4 = addButton("Grenade 4",400,700,183,100,this);

Button bQuitBuyMenuDropMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,800,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lDropWeapon1 = addLabel("",337,100,62,30,this);

Label lDropWeapon2 = addLabel("",337,200,62,30,this);

Label lDropZeus = addLabel("",337,300,62,30,this);

Label lDropGrenade1 = addLabel("",337,400,62,30,this);

Label lDropGrenade2 = addLabel("",337,500,62,30,this);

Label lDropGrenade3 = addLabel("",337,600,62,30,this);

Label lDropGrenade4 = addLabel("",337,700,62,30,this);

int RoundCountbuyMenuDrop;

int BalancebuyMenuDrop;

int RoundCTbuyMenuDrop;

int RoundTbuyMenuDrop;

String buyMenuDropWeapon1;

String buyMenuDropWeapon2;

String buyMenuDropZeus;

String buyMenuDropGrenade1;

String buyMenuDropGrenade2;

String buyMenuDropGrenade3;

String buyMenuDropGrenade4;

public GameTDropMenu(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountBuyMenuDropMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountBuyMenuDropMenu.next() ){

String sqlRoundCountstringBuyMenuDropMenu = sqlRoundCountBuyMenuDropMenu.getString(1);

RoundCountbuyMenuDrop = Integer.parseInt (sqlRoundCountstringBuyMenuDropMenu);}

ResultSet rsBalanceBuyMenuDropMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalanceBuyMenuDropMenu.next() ){

String rs1BalancestringBuyMenuDropMenu = rsBalanceBuyMenuDropMenu.getString(1);

BalancebuyMenuDrop = Integer.parseInt (rs1BalancestringBuyMenuDropMenu);}

ResultSet sqlRoundCTBuyMenuDropMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTBuyMenuDropMenu.next() ){

String sqlRoundCTstringBuyMenuDropMenu = sqlRoundCTBuyMenuDropMenu.getString(1);

RoundCTbuyMenuDrop = Integer.parseInt (sqlRoundCTstringBuyMenuDropMenu);}

ResultSet sqlRoundTBuyMenuDropMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTBuyMenuDropMenu .next() ){

String sqlRoundTstringBuyMenuDropMenu = sqlRoundTBuyMenuDropMenu .getString(1);

RoundTbuyMenuDrop = Integer.parseInt (sqlRoundTstringBuyMenuDropMenu);}

ResultSet sqlWeapon1BuyMenuDropMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1BuyMenuDropMenu .next() ){

buyMenuDropWeapon1 = sqlWeapon1BuyMenuDropMenu .getString(1);

}

ResultSet sqlWeapon2BuyMenuDropMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2BuyMenuDropMenu .next() ){

buyMenuDropWeapon2 = sqlWeapon2BuyMenuDropMenu .getString(1);

}

ResultSet sqlZeusBuyMenuDropMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeusBuyMenuDropMenu .next() ){

buyMenuDropZeus = sqlZeusBuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade1BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1BuyMenuDropMenu .next() ){

buyMenuDropGrenade1 = sqlGrenade1BuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade2BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2BuyMenuDropMenu .next() ){

buyMenuDropGrenade2 = sqlGrenade2BuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade3BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3BuyMenuDropMenu .next() ){

buyMenuDropGrenade3 = sqlGrenade3BuyMenuDropMenu .getString(1);

}

ResultSet sqlGrenade4BuyMenuDropMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4BuyMenuDropMenu .next() ){

buyMenuDropGrenade4 = sqlGrenade4BuyMenuDropMenu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountbuyMenuDrop + "");

tBalance.setText(BalancebuyMenuDrop + "$");

tCTscore.setText(RoundCTbuyMenuDrop + "");

tTscore.setText(RoundTbuyMenuDrop + "");

lDropWeapon1.setText(buyMenuDropWeapon1);

lDropWeapon2.setText(buyMenuDropWeapon2);

lDropZeus.setText(buyMenuDropZeus);

lDropGrenade1.setText(buyMenuDropGrenade1);

lDropGrenade2.setText(buyMenuDropGrenade2);

lDropGrenade3.setText(buyMenuDropGrenade3);

lDropGrenade4.setText(buyMenuDropGrenade4);

if(buyMenuDropWeapon1 == null){

bDropWeapon1.setVisible(false);

}

if(buyMenuDropWeapon2 == null){

bDropWeapon2.setVisible(false);

}

if(buyMenuDropZeus == null){

bDropZeus.setVisible(false);

}

if(buyMenuDropGrenade1 == null){

bDropGrenade1.setVisible(false);

}

if(buyMenuDropGrenade2 == null){

bDropGrenade2.setVisible(false);

}

if(buyMenuDropGrenade3 == null){

bDropGrenade3.setVisible(false);

}

if(buyMenuDropGrenade4 == null){

bDropGrenade4.setVisible(false);

}

}

public void actions(Object source,String command){

if (source == bDropWeapon1){

bDropWeapon1.setEnabled(false);

buyMenuDropWeapon1 = null;

}

if (source == bDropWeapon2){

bDropWeapon2.setEnabled(false);

buyMenuDropWeapon2 = null;

}

if (source == bDropZeus){

bDropZeus.setEnabled(false);

buyMenuDropZeus = null;

}

if (source == bDropGrenade1){

bDropGrenade1.setEnabled(false);

buyMenuDropGrenade1 = null;

}

if (source == bDropGrenade2){

bDropGrenade2.setEnabled(false);

buyMenuDropGrenade2 = null;

}

if (source == bDropGrenade3){

bDropGrenade3.setEnabled(false);

buyMenuDropGrenade3 = null;

}

if (source == bDropGrenade4){

bDropGrenade4.setEnabled(false);

buyMenuDropGrenade4 = null;

}

if (source == bQuitBuyMenuDropMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,buyMenuDropWeapon1);

preparedStatement.setString(2,buyMenuDropWeapon2);

preparedStatement.setString(3,buyMenuDropGrenade1);

preparedStatement.setString(4,buyMenuDropGrenade2);

preparedStatement.setString(5,buyMenuDropGrenade3);

preparedStatement.setString(6,buyMenuDropGrenade4);

preparedStatement.setString(7,buyMenuDropZeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new GameTPickDrop();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class GameTPickMenu extends EasyApp

{

public static void main(String[] args)

{ new GameTPickMenu();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

TextField tWeapon1 = addTextField("",370,600,150,30,this);

TextField tWeapon2 = addTextField("",370,635,150,30,this);

TextField tZeus = addTextField("",370,670,150,30,this);

TextField tGrenade1 = addTextField("",370,705,150,30,this);

TextField tGrenade2 = addTextField("",370,740,150,30,this);

TextField tGrenade3 = addTextField("",370,775,150,30,this);

TextField tGrenade4 = addTextField("",370,810,150,30,this);

TextField tKits = addTextField("",370,845,150,30,this);

Label lGrenadeLabelPickMenu = addLabel("Grenades",70,90,60,30,this);

Label lPistolsLabelPickMenu = addLabel("Pistols",254,90,60,30,this);

Label lZeusLabelPickMenu = addLabel("Equipment",476,90,60,30,this);

Label lSMGLabelPickMenu = addLabel("SMG",742,90,60,30,this);

Label lRiflesLabelPickMenu = addLabel("Rifles",1008,90,60,30,this);

Label lHeavyLabelPickMenu = addLabel("Heavy",1274,90,60,30,this);

Label lYourEquipment = addLabel("Your equipment:",370,570,150,30,this);

Button bMolotov = addButton("Molotov",0,120,183,100, this);

Button bDecoy = addButton("Decoy",0,220,183,100, this);

Button bFlashbang = addButton("Flashbang",0,320,183,100,this);

Button bHEGrenade = addButton("HE Grenade",0,420,183,100,this);

Button bSmoke = addButton("Smoke",0,520,183,100,this);

Button bIncendiary = addButton("Incendiary Grenade",0,620,183,100,this);

Button bGlock18 = addButton("Glock-18",183,120,183,50,this);

Button bDualBerettas= addButton("Dual Berettas",183,170,183,50,this);

Button bP250 = addButton("P250",183,220,183,50,this);

Button bCZ75Auto = addButton("CZ75 - Auto",183,270,183,50,this);

Button bDesertEagle = addButton("Desert Eagle",183,320,183,50,this);

Button bR8Revolver = addButton("R8 Revolver",183,370,183,50,this);

Button bTec9 = addButton("Tec-9",183,420,183,50,this);

Button bUSPS = addButton("USP-S",183,470,183,50,this);

Button bP2000 = addButton("P2000",183,520,183,50,this);

Button bFiveSeveN = addButton("Five-SeveN",183,570,183,50,this);

Button bZeusx27 = addButton("Zeus x27",366,120,266,50,this);

Button bKits = addButton("Kits",366,170,266,50,this);

Button bMAC10 = addButton("MAC-10",632,120,266,50,this);

Button bMP5SD = addButton("MP5-SD",632,170,266,50,this);

Button bMP7 = addButton("MP7",632,220,266,50,this);

Button bUMP45 = addButton("UMP - 45",632,270,266,50,this);

Button bP90 = addButton("P90",632,320,266,50,this);

Button bPPBizon = addButton("PP-Bizon",632,370,266,50,this);

Button bMP9 = addButton("MP9",632,420,266,50,this);

Button bGalilAR = addButton("Galil AR",898,120,266,50,this);

Button bAK47 = addButton("AK-47",898,170,266,50,this);

Button bSSG08 = addButton("SSG 08",898,220,266,50,this);

Button bSG553 = addButton("SG 553",898,270,266,50,this);

Button bAWP = addButton("AWP",898,320,266,50,this);

Button bG3SG1 = addButton("G3SG1",898,370,266,50,this);

Button bFAMAS = addButton("FAMAS",898,420,266,50,this);

Button bM4A4 = addButton("M4A4",898,470,266,50,this);

Button bM4A1S = addButton("M4A1-S",898,520,266,50,this);

Button bAUG = addButton("AUG",898,570,266,50,this);

Button bSCAR20 = addButton("SCAR-20",898,620,266,50,this);

Button bNova = addButton("Nova",1164,120,266,50,this);

Button bXM1014 = addButton("XM1014",1164,170,266,50,this);

Button bSawedOff = addButton("Sawed-Off",1164,220,266,50,this);

Button bM249 = addButton("M249",1164,270,266,50,this);

Button bNegev= addButton("Negev",1164,320,266,50,this);

Button bMAG7= addButton("MAG-7",1164,370,266,50,this);

Button bQuitbuyMenuPickMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,20,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountbuyMenuPick;

int BalancebuyMenuPick;

int RoundCTbuyMenuPick;

int RoundTbuyMenuPick;

int flashbangCountPick = 0;

String buyMenuPickWeapon1;

String buyMenuPickWeapon2;

String buyMenuPickZeus;

String buyMenuPickGrenade1;

String buyMenuPickGrenade2;

String buyMenuPickGrenade3;

String buyMenuPickGrenade4;

String sqlStringLimitGrenadeConditionPickMenu;

String sqlStringSidePickMenu;

String buyMenuPickKits;

public GameTPickMenu(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountbuyMenuPickMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountbuyMenuPickMenu.next() ){

String sqlRoundCountstringbuyMenuPickMenu = sqlRoundCountbuyMenuPickMenu.getString(1);

RoundCountbuyMenuPick = Integer.parseInt (sqlRoundCountstringbuyMenuPickMenu);}

ResultSet rsBalancebuyMenuPickMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancebuyMenuPickMenu.next() ){

String rs1BalancestringbuyMenuPickMenu = rsBalancebuyMenuPickMenu.getString(1);

BalancebuyMenuPick = Integer.parseInt (rs1BalancestringbuyMenuPickMenu);}

ResultSet sqlRoundCTbuyMenuPickMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTbuyMenuPickMenu.next() ){

String sqlRoundCTstringbuyMenuPickMenu = sqlRoundCTbuyMenuPickMenu.getString(1);

RoundCTbuyMenuPick = Integer.parseInt (sqlRoundCTstringbuyMenuPickMenu);}

ResultSet sqlRoundTbuyMenuPickMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTbuyMenuPickMenu .next() ){

String sqlRoundTstringbuyMenuPickMenu = sqlRoundTbuyMenuPickMenu .getString(1);

RoundTbuyMenuPick = Integer.parseInt (sqlRoundTstringbuyMenuPickMenu);}

ResultSet sqlWeapon1buyMenuPickMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1buyMenuPickMenu .next() ){

buyMenuPickWeapon1 = sqlWeapon1buyMenuPickMenu .getString(1);

}

ResultSet sqlWeapon2buyMenuPickMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2buyMenuPickMenu .next() ){

buyMenuPickWeapon2 = sqlWeapon2buyMenuPickMenu .getString(1);

}

ResultSet sqlZeusbuyMenuPickMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeusbuyMenuPickMenu .next() ){

buyMenuPickZeus = sqlZeusbuyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade1buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1buyMenuPickMenu .next() ){

buyMenuPickGrenade1 = sqlGrenade1buyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade2buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2buyMenuPickMenu .next() ){

buyMenuPickGrenade2 = sqlGrenade2buyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade3buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3buyMenuPickMenu .next() ){

buyMenuPickGrenade3 = sqlGrenade3buyMenuPickMenu .getString(1);

}

ResultSet sqlGrenade4buyMenuPickMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4buyMenuPickMenu .next() ){

buyMenuPickGrenade4 = sqlGrenade4buyMenuPickMenu .getString(1);

}

ResultSet sqlLimitGrenadeConditionPickMenu = stmt.executeQuery( "SELECT LimitGrenadeCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlLimitGrenadeConditionPickMenu.next() ){

sqlStringLimitGrenadeConditionPickMenu = sqlLimitGrenadeConditionPickMenu.getString(1);}

ResultSet sqlSidePickMenu = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlSidePickMenu.next() ){

sqlStringSidePickMenu = sqlSidePickMenu.getString(1);}

ResultSet kitsPistolsSQL = stmt.executeQuery( "SELECT kits FROM IACompCSGO WHERE idNumCS = 1;") ;

while( kitsPistolsSQL.next() ){

buyMenuPickKits = kitsPistolsSQL.getString(1);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountbuyMenuPick + "");

tBalance.setText(BalancebuyMenuPick + "$");

tCTscore.setText(RoundCTbuyMenuPick + "");

tTscore.setText(RoundTbuyMenuPick + "");

tWeapon1.setText(buyMenuPickWeapon1 + "");

tWeapon2.setText(buyMenuPickWeapon2 + "");

tZeus.setText(buyMenuPickZeus + "");

tGrenade1.setText(buyMenuPickGrenade1 + "");

tGrenade2.setText(buyMenuPickGrenade2 + "");

tGrenade3.setText(buyMenuPickGrenade3 + "");

tGrenade4.setText(buyMenuPickGrenade4 + "");

tKits.setText(buyMenuPickKits + "");

if((RoundCountbuyMenuPick == 1 || RoundCountbuyMenuPick == 16) && sqlStringSidePickMenu.equals("TSide")){

bMP9.setEnabled(false);

bFAMAS.setEnabled(false);

bM4A4.setEnabled(false);

bM4A1S.setEnabled(false);

bAUG.setEnabled(false);

bSCAR20.setEnabled(false);

bMAG7.setEnabled(false);

bMAC10.setEnabled(false);

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

bUMP45.setEnabled(false);

bP90.setEnabled(false);

bPPBizon.setEnabled(false);

bMP9.setEnabled(false);

bGalilAR.setEnabled(false);

bAK47.setEnabled(false);

bSSG08.setEnabled(false);

bSG553.setEnabled(false);

bAWP.setEnabled(false);

bG3SG1.setEnabled(false);

bNova.setEnabled(false);

bXM1014.setEnabled(false);

bSawedOff.setEnabled(false);

bM249.setEnabled(false);

bNegev.setEnabled(false);

}

if((RoundCountbuyMenuPick == 1 || RoundCountbuyMenuPick == 16) && sqlStringSidePickMenu.equals("CTSide")){

bMP9.setEnabled(false);

bFAMAS.setEnabled(false);

bM4A4.setEnabled(false);

bM4A1S.setEnabled(false);

bAUG.setEnabled(false);

bSCAR20.setEnabled(false);

bMAG7.setEnabled(false);

bMAC10.setEnabled(false);

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

bUMP45.setEnabled(false);

bP90.setEnabled(false);

bPPBizon.setEnabled(false);

bMP9.setEnabled(false);

bGalilAR.setEnabled(false);

bAK47.setEnabled(false);

bSSG08.setEnabled(false);

bSG553.setEnabled(false);

bAWP.setEnabled(false);

bG3SG1.setEnabled(false);

bNova.setEnabled(false);

bXM1014.setEnabled(false);

bSawedOff.setEnabled(false);

bM249.setEnabled(false);

bNegev.setEnabled(false);

}

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Flashbang")){

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade1.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade1.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Flashbang")){

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade2.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade2.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Flashbang")){

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade3.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade3.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickGrenade4 != null){

if(buyMenuPickGrenade4.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Flashbang")){

if(buyMenuPickGrenade1 != null){

if(buyMenuPickGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade2 != null){

if(buyMenuPickGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(buyMenuPickGrenade3 != null){

if(buyMenuPickGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(buyMenuPickGrenade4.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(buyMenuPickGrenade4.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(buyMenuPickKits != null){

if(buyMenuPickKits.equals("Kits")){

bKits.setEnabled(false);

}

}

if(sqlStringSidePickMenu.equals("TSide")){

bKits.setVisible(false);

}

}

public void actions(Object source,String command){

//grenades

if(source == bMolotov){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Molotov";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Molotov";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Molotov";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Molotov";

tGrenade4.setText(buyMenuPickGrenade4);

}

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bDecoy){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Decoy";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Decoy";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Decoy";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Decoy";

tGrenade4.setText(buyMenuPickGrenade4);

}

bDecoy.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bFlashbang){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Flashbang";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Flashbang";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Flashbang";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Flashbang";

tGrenade4.setText(buyMenuPickGrenade4);

}

flashbangCountPick = flashbangCountPick + 1;

if(flashbangCountPick == 2){

bFlashbang.setEnabled(false);

}

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bHEGrenade){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "HE Grenade";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "HE Grenade";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "HE Grenade";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "HE Grenade";

tGrenade4.setText(buyMenuPickGrenade4);

}

bHEGrenade.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bSmoke){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Smoke";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Smoke";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Smoke";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Smoke";

tGrenade4.setText(buyMenuPickGrenade4);

}

bSmoke.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bIncendiary){

if(buyMenuPickGrenade1 == null){

buyMenuPickGrenade1 = "Incendiary Grenade";

tGrenade1.setText(buyMenuPickGrenade1);

}else if(buyMenuPickGrenade2 == null){

buyMenuPickGrenade2 = "Incendiary Grenade";

tGrenade2.setText(buyMenuPickGrenade2);

}else if(buyMenuPickGrenade3 == null){

buyMenuPickGrenade3 = "Incendiary Grenade";

tGrenade3.setText(buyMenuPickGrenade3);

}else if(buyMenuPickGrenade4 == null){

buyMenuPickGrenade4 = "Incendiary Grenade";

tGrenade4.setText(buyMenuPickGrenade4);

}

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

if(buyMenuPickGrenade1 != null && buyMenuPickGrenade2 != null && buyMenuPickGrenade3 != null && buyMenuPickGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

//pistols

if (source == bGlock18){

buyMenuPickWeapon1 = "Glock-18";

tWeapon1.setText(buyMenuPickWeapon1);

}

if (source == bDualBerettas){

buyMenuPickWeapon1 = "Dual Berettas";

tWeapon1.setText(buyMenuPickWeapon1);

}

if (source == bP250){

buyMenuPickWeapon1 = "P250";

tWeapon1.setText(buyMenuPickWeapon1);

}

if (source == bCZ75Auto){

buyMenuPickWeapon1 = "CZ75-Auto";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bDesertEagle){

buyMenuPickWeapon1 = "Desert Eagle";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bR8Revolver){

buyMenuPickWeapon1 = "R8 Revolver";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bTec9 ){

buyMenuPickWeapon1 = "Tec-9";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bUSPS){

buyMenuPickWeapon1 = "USP-S";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bP2000){

buyMenuPickWeapon1 = "P2000";

tWeapon1.setText(buyMenuPickWeapon1);

}

if(source == bFiveSeveN){

buyMenuPickWeapon1 = "Five-SeveN";

tWeapon1.setText(buyMenuPickWeapon1);

}

//zeus

if(source == bZeusx27){

buyMenuPickZeus = "Zeus";

tZeus.setText(buyMenuPickZeus);

}

if(source == bKits){

buyMenuPickKits = "Kits";

tKits.setText(buyMenuPickKits);

}

//smgs

if(source == bMAC10){

buyMenuPickWeapon2 = "MAC-10";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMP5SD){

buyMenuPickWeapon2 = "MP5-SD";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMP7){

buyMenuPickWeapon2 = "MP7";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bUMP45){

buyMenuPickWeapon2 = "UMP-45";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bP90){

buyMenuPickWeapon2 = "P90";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bPPBizon){

buyMenuPickWeapon2 = "PP-Bizon";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMP9){

buyMenuPickWeapon2 = "MP9";

tWeapon2.setText(buyMenuPickWeapon2);

}

//rifles

if(source == bGalilAR){

buyMenuPickWeapon2 = "Galil AR";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bAK47){

buyMenuPickWeapon2 = "AK-47";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSSG08){

buyMenuPickWeapon2 = "SSG 08";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSG553){

buyMenuPickWeapon2 = "SG 553";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bAWP){

buyMenuPickWeapon2 = "AWP";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bG3SG1){

buyMenuPickWeapon2 = "G3SG1";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bFAMAS){

buyMenuPickWeapon2 = "FAMAS";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bM4A4){

buyMenuPickWeapon2 = "M4A4";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bM4A1S){

buyMenuPickWeapon2 = "M4A1-S";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bAUG){

buyMenuPickWeapon2 = "AUG";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSCAR20){

buyMenuPickWeapon2 = "SCAR-20";

tWeapon2.setText(buyMenuPickWeapon2);

}

//heavy

if(source == bNova){

buyMenuPickWeapon2 = "Nova";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bXM1014){

buyMenuPickWeapon2 = "XM1014";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bSawedOff){

buyMenuPickWeapon2 = "Sawed-Off";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bM249){

buyMenuPickWeapon2 = "M249";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bNegev){

buyMenuPickWeapon2 = "Negev";

tWeapon2.setText(buyMenuPickWeapon2);

}

if(source == bMAG7){

buyMenuPickWeapon2 = "MAG-7";

tWeapon2.setText(buyMenuPickWeapon2);

}

if (source == bQuitbuyMenuPickMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ?, kits = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,buyMenuPickWeapon1);

preparedStatement.setString(2,buyMenuPickWeapon2);

preparedStatement.setString(3,buyMenuPickGrenade1);

preparedStatement.setString(4,buyMenuPickGrenade2);

preparedStatement.setString(5,buyMenuPickGrenade3);

preparedStatement.setString(6,buyMenuPickGrenade4);

preparedStatement.setString(7,buyMenuPickZeus);

preparedStatement.setString(8,buyMenuPickKits);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new GameTPickDrop();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class killMenuTPickDrop extends EasyApp

{

public static void main(String[] args)

{ new killMenuTPickDrop();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropChoice = addButton("Did you drop a weapon?",100,300,798,100,this);

Button bPickChoice = addButton("Did you pick a weapon?",100,400,798,100,this);

Button bQuitKillMenuTPickDrop = addButton("Go Back",850,600,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountkillMenuTPickDrop;

int BalancekillMenuTPickDrop;

int RoundCTkillMenuTPickDrop;

int RoundTkillMenuTPickDrop;

public killMenuTPickDrop(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuTPickDrop = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuTPickDrop.next() ){

String sqlRoundCountstringkillMenuTPickDrop = sqlRoundCountkillMenuTPickDrop.getString(1);

RoundCountkillMenuTPickDrop = Integer.parseInt (sqlRoundCountstringkillMenuTPickDrop);}

ResultSet rsBalancekillMenuTPickDrop = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuTPickDrop.next() ){

String rs1BalancestringkillMenuTPickDrop = rsBalancekillMenuTPickDrop.getString(1);

BalancekillMenuTPickDrop = Integer.parseInt (rs1BalancestringkillMenuTPickDrop);}

ResultSet sqlRoundCTkillMenuTPickDrop = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuTPickDrop.next() ){

String sqlRoundCTstringkillMenuTPickDrop = sqlRoundCTkillMenuTPickDrop.getString(1);

RoundCTkillMenuTPickDrop = Integer.parseInt (sqlRoundCTstringkillMenuTPickDrop);}

ResultSet sqlRoundTkillMenuTPickDrop = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuTPickDrop .next() ){

String sqlRoundTstringkillMenuTPickDrop = sqlRoundTkillMenuTPickDrop .getString(1);

RoundTkillMenuTPickDrop = Integer.parseInt (sqlRoundTstringkillMenuTPickDrop);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,700);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuTPickDrop + "");

tBalance.setText(BalancekillMenuTPickDrop + "$");

tCTscore.setText(RoundCTkillMenuTPickDrop + "");

tTscore.setText(RoundTkillMenuTPickDrop + "");

}

public void actions(Object source,String command){

if (source == bQuitKillMenuTPickDrop )

{

new killMenuT();

dispose();

}

if (source == bDropChoice)

{

new killMenuTDropMenu();

dispose();

}

if (source == bPickChoice)

{

new killMenuTPickMenu();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class killMenuTDropMenu extends EasyApp

{

public static void main(String[] args)

{ new killMenuTDropMenu();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,800,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bDropWeapon1 = addButton("Secondary weapon",400,100,183,100,this);

Button bDropWeapon2 = addButton("Primary weapon",400,200,183,100,this);

Button bDropZeus = addButton("Zeus",400,300,183,100,this);

Button bDropGrenade1 = addButton("Grenade 1",400,400,183,100,this);

Button bDropGrenade2 = addButton("Grenade 2",400,500,183,100,this);

Button bDropGrenade3 = addButton("Grenade 3",400,600,183,100,this);

Button bDropGrenade4 = addButton("Grenade 4",400,700,183,100,this);

Button bQuitkillMenuTDropMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,800,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lDropWeapon1 = addLabel("",337,100,62,30,this);

Label lDropWeapon2 = addLabel("",337,200,62,30,this);

Label lDropZeus = addLabel("",337,300,62,30,this);

Label lDropGrenade1 = addLabel("",337,400,62,30,this);

Label lDropGrenade2 = addLabel("",337,500,62,30,this);

Label lDropGrenade3 = addLabel("",337,600,62,30,this);

Label lDropGrenade4 = addLabel("",337,700,62,30,this);

int RoundCountkillMenuTDropMenu;

int BalancekillMenuTDropMenu;

int RoundCTkillMenuTDropMenu;

int RoundTkillMenuTDropMenu;

String killMenuTDropMenuWeapon1;

String killMenuTDropMenuWeapon2;

String killMenuTDropMenuZeus;

String killMenuTDropMenuGrenade1;

String killMenuTDropMenuGrenade2;

String killMenuTDropMenuGrenade3;

String killMenuTDropMenuGrenade4;

public killMenuTDropMenu(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuTDropMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuTDropMenu.next() ){

String sqlRoundCountstringkillMenuTDropMenu = sqlRoundCountkillMenuTDropMenu.getString(1);

RoundCountkillMenuTDropMenu = Integer.parseInt (sqlRoundCountstringkillMenuTDropMenu);}

ResultSet rsBalancekillMenuTDropMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuTDropMenu.next() ){

String rs1BalancestringkillMenuTDropMenu = rsBalancekillMenuTDropMenu.getString(1);

BalancekillMenuTDropMenu = Integer.parseInt (rs1BalancestringkillMenuTDropMenu);}

ResultSet sqlRoundCTkillMenuTDropMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuTDropMenu.next() ){

String sqlRoundCTstringkillMenuTDropMenu = sqlRoundCTkillMenuTDropMenu.getString(1);

RoundCTkillMenuTDropMenu = Integer.parseInt (sqlRoundCTstringkillMenuTDropMenu);}

ResultSet sqlRoundTkillMenuTDropMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuTDropMenu .next() ){

String sqlRoundTstringkillMenuTDropMenu = sqlRoundTkillMenuTDropMenu .getString(1);

RoundTkillMenuTDropMenu = Integer.parseInt (sqlRoundTstringkillMenuTDropMenu);}

ResultSet sqlWeapon1killMenuTDropMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuTDropMenu .next() ){

killMenuTDropMenuWeapon1 = sqlWeapon1killMenuTDropMenu .getString(1);

}

ResultSet sqlWeapon2killMenuTDropMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuTDropMenu .next() ){

killMenuTDropMenuWeapon2 = sqlWeapon2killMenuTDropMenu .getString(1);

}

ResultSet sqlZeuskillMenuTDropMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuTDropMenu .next() ){

killMenuTDropMenuZeus = sqlZeuskillMenuTDropMenu .getString(1);

}

ResultSet sqlGrenade1killMenuTDropMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuTDropMenu .next() ){

killMenuTDropMenuGrenade1 = sqlGrenade1killMenuTDropMenu .getString(1);

}

ResultSet sqlGrenade2killMenuTDropMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuTDropMenu .next() ){

killMenuTDropMenuGrenade2 = sqlGrenade2killMenuTDropMenu .getString(1);

}

ResultSet sqlGrenade3killMenuTDropMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuTDropMenu .next() ){

killMenuTDropMenuGrenade3 = sqlGrenade3killMenuTDropMenu .getString(1);

}

ResultSet sqlGrenade4killMenuTDropMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuTDropMenu .next() ){

killMenuTDropMenuGrenade4 = sqlGrenade4killMenuTDropMenu .getString(1);

}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuTDropMenu + "");

tBalance.setText(BalancekillMenuTDropMenu + "$");

tCTscore.setText(RoundCTkillMenuTDropMenu + "");

tTscore.setText(RoundTkillMenuTDropMenu + "");

lDropWeapon1.setText(killMenuTDropMenuWeapon1);

lDropWeapon2.setText(killMenuTDropMenuWeapon2);

lDropZeus.setText(killMenuTDropMenuZeus);

lDropGrenade1.setText(killMenuTDropMenuGrenade1);

lDropGrenade2.setText(killMenuTDropMenuGrenade2);

lDropGrenade3.setText(killMenuTDropMenuGrenade3);

lDropGrenade4.setText(killMenuTDropMenuGrenade4);

if(killMenuTDropMenuWeapon1 == null){

bDropWeapon1.setVisible(false);

}

if(killMenuTDropMenuWeapon2 == null){

bDropWeapon2.setVisible(false);

}

if(killMenuTDropMenuZeus == null){

bDropZeus.setVisible(false);

}

if(killMenuTDropMenuGrenade1 == null){

bDropGrenade1.setVisible(false);

}

if(killMenuTDropMenuGrenade2 == null){

bDropGrenade2.setVisible(false);

}

if(killMenuTDropMenuGrenade3 == null){

bDropGrenade3.setVisible(false);

}

if(killMenuTDropMenuGrenade4 == null){

bDropGrenade4.setVisible(false);

}

}

public void actions(Object source,String command){

if (source == bDropWeapon1){

bDropWeapon1.setEnabled(false);

killMenuTDropMenuWeapon1 = null;

}

if (source == bDropWeapon2){

bDropWeapon2.setEnabled(false);

killMenuTDropMenuWeapon2 = null;

}

if (source == bDropZeus){

bDropZeus.setEnabled(false);

killMenuTDropMenuZeus = null;

}

if (source == bDropGrenade1){

bDropGrenade1.setEnabled(false);

killMenuTDropMenuGrenade1 = null;

}

if (source == bDropGrenade2){

bDropGrenade2.setEnabled(false);

killMenuTDropMenuGrenade2 = null;

}

if (source == bDropGrenade3){

bDropGrenade3.setEnabled(false);

killMenuTDropMenuGrenade3 = null;

}

if (source == bDropGrenade4){

bDropGrenade4.setEnabled(false);

killMenuTDropMenuGrenade4 = null;

}

if (source == bQuitkillMenuTDropMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuTDropMenuWeapon1);

preparedStatement.setString(2,killMenuTDropMenuWeapon2);

preparedStatement.setString(3,killMenuTDropMenuGrenade1);

preparedStatement.setString(4,killMenuTDropMenuGrenade2);

preparedStatement.setString(5,killMenuTDropMenuGrenade3);

preparedStatement.setString(6,killMenuTDropMenuGrenade4);

preparedStatement.setString(7,killMenuTDropMenuZeus);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuTPickDrop();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class killMenuTPickMenu extends EasyApp

{

public static void main(String[] args)

{ new killMenuTPickMenu();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,50,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

TextField tWeapon1 = addTextField("",370,600,150,30,this);

TextField tWeapon2 = addTextField("",370,635,150,30,this);

TextField tZeus = addTextField("",370,670,150,30,this);

TextField tGrenade1 = addTextField("",370,705,150,30,this);

TextField tGrenade2 = addTextField("",370,740,150,30,this);

TextField tGrenade3 = addTextField("",370,775,150,30,this);

TextField tGrenade4 = addTextField("",370,810,150,30,this);

TextField tKits = addTextField("",370,845,150,30,this);

Label lGrenadeLabelPickMenu = addLabel("Grenades",70,90,60,30,this);

Label lPistolsLabelPickMenu = addLabel("Pistols",254,90,60,30,this);

Label lZeusLabelPickMenu = addLabel("Zeus",476,90,60,30,this);

Label lSMGLabelPickMenu = addLabel("SMG",742,90,60,30,this);

Label lRiflesLabelPickMenu = addLabel("Rifles",1008,90,60,30,this);

Label lHeavyLabelPickMenu = addLabel("Heavy",1274,90,60,30,this);

Label lYourEquipment = addLabel("Your equipment:",370,570,150,30,this);

Button bMolotov = addButton("Molotov",0,120,183,100, this);

Button bDecoy = addButton("Decoy",0,220,183,100, this);

Button bFlashbang = addButton("Flashbang",0,320,183,100,this);

Button bHEGrenade = addButton("HE Grenade",0,420,183,100,this);

Button bSmoke = addButton("Smoke",0,520,183,100,this);

Button bIncendiary = addButton("Incendiary Grenade",0,620,183,100,this);

Button bGlock18 = addButton("Glock-18",183,120,183,50,this);

Button bDualBerettas= addButton("Dual Berettas",183,170,183,50,this);

Button bP250 = addButton("P250",183,220,183,50,this);

Button bCZ75Auto = addButton("CZ75 - Auto",183,270,183,50,this);

Button bDesertEagle = addButton("Desert Eagle",183,320,183,50,this);

Button bR8Revolver = addButton("R8 Revolver",183,370,183,50,this);

Button bTec9 = addButton("Tec-9",183,420,183,50,this);

Button bUSPS = addButton("USP-S",183,470,183,50,this);

Button bP2000 = addButton("P2000",183,520,183,50,this);

Button bFiveSeveN = addButton("Five-SeveN",183,570,183,50,this);

Button bZeusx27 = addButton("Zeus x27",366,120,266,50,this);

Button bKits = addButton("Kits",366,170,266,50,this);

Button bMAC10 = addButton("MAC-10",632,120,266,50,this);

Button bMP5SD = addButton("MP5-SD",632,170,266,50,this);

Button bMP7 = addButton("MP7",632,220,266,50,this);

Button bUMP45 = addButton("UMP - 45",632,270,266,50,this);

Button bP90 = addButton("P90",632,320,266,50,this);

Button bPPBizon = addButton("PP-Bizon",632,370,266,50,this);

Button bMP9 = addButton("MP9",632,420,266,50,this);

Button bGalilAR = addButton("Galil AR",898,120,266,50,this);

Button bAK47 = addButton("AK-47",898,170,266,50,this);

Button bSSG08 = addButton("SSG 08",898,220,266,50,this);

Button bSG553 = addButton("SG 553",898,270,266,50,this);

Button bAWP = addButton("AWP",898,320,266,50,this);

Button bG3SG1 = addButton("G3SG1",898,370,266,50,this);

Button bFAMAS = addButton("FAMAS",898,420,266,50,this);

Button bM4A4 = addButton("M4A4",898,470,266,50,this);

Button bM4A1S = addButton("M4A1-S",898,520,266,50,this);

Button bAUG = addButton("AUG",898,570,266,50,this);

Button bSCAR20 = addButton("SCAR-20",898,620,266,50,this);

Button bNova = addButton("Nova",1164,120,266,50,this);

Button bXM1014 = addButton("XM1014",1164,170,266,50,this);

Button bSawedOff = addButton("Sawed-Off",1164,220,266,50,this);

Button bM249 = addButton("M249",1164,270,266,50,this);

Button bNegev= addButton("Negev",1164,320,266,50,this);

Button bMAG7= addButton("MAG-7",1164,370,266,50,this);

Button bQuitkillMenuTPickMenu = addButton("Go Back",850,750,150,100,this);

Label lCT = addLabel("CT",330,50,20,30,this);

Label lT = addLabel("T",420,50,10,30,this);

Label lRoundNum = addLabel("Round: ",455,50,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int RoundCountkillMenuTPickMenu;

int BalancekillMenuTPickMenu;

int RoundCTkillMenuTPickMenu;

int RoundTkillMenuTPickMenu;

int flashbangCountPick = 0;

String killMenuTPickMenuWeapon1;

String killMenuTPickMenuWeapon2;

String killMenuTPickMenuZeus;

String killMenuTPickMenuGrenade1;

String killMenuTPickMenuGrenade2;

String killMenuTPickMenuGrenade3;

String killMenuTPickMenuGrenade4;

String killMenuTPickMenuKits;

String sqlStringLimitGrenadeConditionPickMenu;

String sqlStringSidePickMenu;

public killMenuTPickMenu(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountkillMenuTPickMenu = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountkillMenuTPickMenu.next() ){

String sqlRoundCountstringkillMenuTPickMenu = sqlRoundCountkillMenuTPickMenu.getString(1);

RoundCountkillMenuTPickMenu = Integer.parseInt (sqlRoundCountstringkillMenuTPickMenu);}

ResultSet rsBalancekillMenuTPickMenu = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalancekillMenuTPickMenu.next() ){

String rs1BalancestringkillMenuTPickMenu = rsBalancekillMenuTPickMenu.getString(1);

BalancekillMenuTPickMenu = Integer.parseInt (rs1BalancestringkillMenuTPickMenu);}

ResultSet sqlRoundCTkillMenuTPickMenu = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTkillMenuTPickMenu.next() ){

String sqlRoundCTstringkillMenuTPickMenu = sqlRoundCTkillMenuTPickMenu.getString(1);

RoundCTkillMenuTPickMenu = Integer.parseInt (sqlRoundCTstringkillMenuTPickMenu);}

ResultSet sqlRoundTkillMenuTPickMenu = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTkillMenuTPickMenu .next() ){

String sqlRoundTstringkillMenuTPickMenu = sqlRoundTkillMenuTPickMenu .getString(1);

RoundTkillMenuTPickMenu = Integer.parseInt (sqlRoundTstringkillMenuTPickMenu);}

ResultSet sqlWeapon1killMenuTPickMenu = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1killMenuTPickMenu .next() ){

killMenuTPickMenuWeapon1 = sqlWeapon1killMenuTPickMenu .getString(1);

}

ResultSet sqlWeapon2killMenuTPickMenu = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2killMenuTPickMenu .next() ){

killMenuTPickMenuWeapon2 = sqlWeapon2killMenuTPickMenu .getString(1);

}

ResultSet sqlZeuskillMenuTPickMenu = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeuskillMenuTPickMenu .next() ){

killMenuTPickMenuZeus = sqlZeuskillMenuTPickMenu .getString(1);

}

ResultSet sqlGrenade1killMenuTPickMenu = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1killMenuTPickMenu .next() ){

killMenuTPickMenuGrenade1 = sqlGrenade1killMenuTPickMenu .getString(1);

}

ResultSet sqlGrenade2killMenuTPickMenu = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2killMenuTPickMenu .next() ){

killMenuTPickMenuGrenade2 = sqlGrenade2killMenuTPickMenu .getString(1);

}

ResultSet sqlGrenade3killMenuTPickMenu = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3killMenuTPickMenu .next() ){

killMenuTPickMenuGrenade3 = sqlGrenade3killMenuTPickMenu .getString(1);

}

ResultSet sqlGrenade4killMenuTPickMenu = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4killMenuTPickMenu .next() ){

killMenuTPickMenuGrenade4 = sqlGrenade4killMenuTPickMenu .getString(1);

}

ResultSet sqlLimitGrenadeConditionPickMenu = stmt.executeQuery( "SELECT LimitGrenadeCondition FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlLimitGrenadeConditionPickMenu.next() ){

sqlStringLimitGrenadeConditionPickMenu = sqlLimitGrenadeConditionPickMenu.getString(1);}

ResultSet sqlSidePickMenu = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlSidePickMenu.next() ){

sqlStringSidePickMenu = sqlSidePickMenu.getString(1);}

ResultSet kitsPistolsSQL = stmt.executeQuery( "SELECT kits FROM IACompCSGO WHERE idNumCS = 1;") ;

while( kitsPistolsSQL.next() ){

killMenuTPickMenuKits = kitsPistolsSQL.getString(1);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountkillMenuTPickMenu + "");

tBalance.setText(BalancekillMenuTPickMenu + "$");

tCTscore.setText(RoundCTkillMenuTPickMenu + "");

tTscore.setText(RoundTkillMenuTPickMenu + "");

tWeapon1.setText(killMenuTPickMenuWeapon1 + "");

tWeapon2.setText(killMenuTPickMenuWeapon2 + "");

tZeus.setText(killMenuTPickMenuZeus + "");

tGrenade1.setText(killMenuTPickMenuGrenade1 + "");

tGrenade2.setText(killMenuTPickMenuGrenade2 + "");

tGrenade3.setText(killMenuTPickMenuGrenade3 + "");

tGrenade4.setText(killMenuTPickMenuGrenade4 + "");

tKits.setText(killMenuTPickMenuKits + "");

if((RoundCountkillMenuTPickMenu == 1 || RoundCountkillMenuTPickMenu == 16) && sqlStringSidePickMenu.equals("TSide")){

bMP9.setEnabled(false);

bFAMAS.setEnabled(false);

bM4A4.setEnabled(false);

bM4A1S.setEnabled(false);

bAUG.setEnabled(false);

bSCAR20.setEnabled(false);

bMAG7.setEnabled(false);

bMAC10.setEnabled(false);

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

bUMP45.setEnabled(false);

bP90.setEnabled(false);

bPPBizon.setEnabled(false);

bMP9.setEnabled(false);

bGalilAR.setEnabled(false);

bAK47.setEnabled(false);

bSSG08.setEnabled(false);

bSG553.setEnabled(false);

bAWP.setEnabled(false);

bG3SG1.setEnabled(false);

bNova.setEnabled(false);

bXM1014.setEnabled(false);

bSawedOff.setEnabled(false);

bM249.setEnabled(false);

bNegev.setEnabled(false);

}

if((RoundCountkillMenuTPickMenu == 1 || RoundCountkillMenuTPickMenu == 16) && sqlStringSidePickMenu.equals("CTSide")){

bMP9.setEnabled(false);

bFAMAS.setEnabled(false);

bM4A4.setEnabled(false);

bM4A1S.setEnabled(false);

bAUG.setEnabled(false);

bSCAR20.setEnabled(false);

bMAG7.setEnabled(false);

bMAC10.setEnabled(false);

bMP5SD.setEnabled(false);

bMP7.setEnabled(false);

bUMP45.setEnabled(false);

bP90.setEnabled(false);

bPPBizon.setEnabled(false);

bMP9.setEnabled(false);

bGalilAR.setEnabled(false);

bAK47.setEnabled(false);

bSSG08.setEnabled(false);

bSG553.setEnabled(false);

bAWP.setEnabled(false);

bG3SG1.setEnabled(false);

bNova.setEnabled(false);

bXM1014.setEnabled(false);

bSawedOff.setEnabled(false);

bM249.setEnabled(false);

bNegev.setEnabled(false);

}

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(killMenuTPickMenuGrenade1 != null){

if(killMenuTPickMenuGrenade1.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(killMenuTPickMenuGrenade1.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(killMenuTPickMenuGrenade1.equals("Flashbang")){

if(killMenuTPickMenuGrenade2 != null){

if(killMenuTPickMenuGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade3 != null){

if(killMenuTPickMenuGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade4 != null){

if(killMenuTPickMenuGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(killMenuTPickMenuGrenade1.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(killMenuTPickMenuGrenade1.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(killMenuTPickMenuGrenade1.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade2 != null){

if(killMenuTPickMenuGrenade2.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(killMenuTPickMenuGrenade2.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(killMenuTPickMenuGrenade2.equals("Flashbang")){

if(killMenuTPickMenuGrenade1 != null){

if(killMenuTPickMenuGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade3 != null){

if(killMenuTPickMenuGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade4 != null){

if(killMenuTPickMenuGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(killMenuTPickMenuGrenade2.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(killMenuTPickMenuGrenade2.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(killMenuTPickMenuGrenade2.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade3 != null){

if(killMenuTPickMenuGrenade3.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(killMenuTPickMenuGrenade3.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(killMenuTPickMenuGrenade3.equals("Flashbang")){

if(killMenuTPickMenuGrenade1 != null){

if(killMenuTPickMenuGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade2 != null){

if(killMenuTPickMenuGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade4 != null){

if(killMenuTPickMenuGrenade4.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(killMenuTPickMenuGrenade3.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(killMenuTPickMenuGrenade3.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(killMenuTPickMenuGrenade3.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade4 != null){

if(killMenuTPickMenuGrenade4.equals("Molotov")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

if(killMenuTPickMenuGrenade4.equals("Decoy")){

bDecoy.setEnabled(false);

}

if(killMenuTPickMenuGrenade4.equals("Flashbang")){

if(killMenuTPickMenuGrenade1 != null){

if(killMenuTPickMenuGrenade1.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade2 != null){

if(killMenuTPickMenuGrenade2.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

if(killMenuTPickMenuGrenade3 != null){

if(killMenuTPickMenuGrenade3.equals("Flashbang")){

bFlashbang.setEnabled(false);

}

}

}

if(killMenuTPickMenuGrenade4.equals("HE Grenade")){

bHEGrenade.setEnabled(false);

}

if(killMenuTPickMenuGrenade4.equals("Smoke")){

bSmoke.setEnabled(false);

}

if(killMenuTPickMenuGrenade4.equals("Incendiary Grenade")){

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(killMenuTPickMenuKits != null){

if(killMenuTPickMenuKits.equals("Kits")){

bKits.setEnabled(false);

}

}

if(sqlStringSidePickMenu.equals("TSide")){

bKits.setVisible(false);

}

}

public void actions(Object source,String command){

//grenades

if(source == bMolotov){

if(killMenuTPickMenuGrenade1 == null){

killMenuTPickMenuGrenade1 = "Molotov";

tGrenade1.setText(killMenuTPickMenuGrenade1);

}else if(killMenuTPickMenuGrenade2 == null){

killMenuTPickMenuGrenade2 = "Molotov";

tGrenade2.setText(killMenuTPickMenuGrenade2);

}else if(killMenuTPickMenuGrenade3 == null){

killMenuTPickMenuGrenade3 = "Molotov";

tGrenade3.setText(killMenuTPickMenuGrenade3);

}else if(killMenuTPickMenuGrenade4 == null){

killMenuTPickMenuGrenade4 = "Molotov";

tGrenade4.setText(killMenuTPickMenuGrenade4);

}

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bDecoy){

if(killMenuTPickMenuGrenade1 == null){

killMenuTPickMenuGrenade1 = "Decoy";

tGrenade1.setText(killMenuTPickMenuGrenade1);

}else if(killMenuTPickMenuGrenade2 == null){

killMenuTPickMenuGrenade2 = "Decoy";

tGrenade2.setText(killMenuTPickMenuGrenade2);

}else if(killMenuTPickMenuGrenade3 == null){

killMenuTPickMenuGrenade3 = "Decoy";

tGrenade3.setText(killMenuTPickMenuGrenade3);

}else if(killMenuTPickMenuGrenade4 == null){

killMenuTPickMenuGrenade4 = "Decoy";

tGrenade4.setText(killMenuTPickMenuGrenade4);

}

bDecoy.setEnabled(false);

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bFlashbang){

if(killMenuTPickMenuGrenade1 == null){

killMenuTPickMenuGrenade1 = "Flashbang";

tGrenade1.setText(killMenuTPickMenuGrenade1);

}else if(killMenuTPickMenuGrenade2 == null){

killMenuTPickMenuGrenade2 = "Flashbang";

tGrenade2.setText(killMenuTPickMenuGrenade2);

}else if(killMenuTPickMenuGrenade3 == null){

killMenuTPickMenuGrenade3 = "Flashbang";

tGrenade3.setText(killMenuTPickMenuGrenade3);

}else if(killMenuTPickMenuGrenade4 == null){

killMenuTPickMenuGrenade4 = "Flashbang";

tGrenade4.setText(killMenuTPickMenuGrenade4);

}

flashbangCountPick = flashbangCountPick + 1;

if(flashbangCountPick == 2){

bFlashbang.setEnabled(false);

}

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bHEGrenade){

if(killMenuTPickMenuGrenade1 == null){

killMenuTPickMenuGrenade1 = "HE Grenade";

tGrenade1.setText(killMenuTPickMenuGrenade1);

}else if(killMenuTPickMenuGrenade2 == null){

killMenuTPickMenuGrenade2 = "HE Grenade";

tGrenade2.setText(killMenuTPickMenuGrenade2);

}else if(killMenuTPickMenuGrenade3 == null){

killMenuTPickMenuGrenade3 = "HE Grenade";

tGrenade3.setText(killMenuTPickMenuGrenade3);

}else if(killMenuTPickMenuGrenade4 == null){

killMenuTPickMenuGrenade4 = "HE Grenade";

tGrenade4.setText(killMenuTPickMenuGrenade4);

}

bHEGrenade.setEnabled(false);

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bSmoke){

if(killMenuTPickMenuGrenade1 == null){

killMenuTPickMenuGrenade1 = "Smoke";

tGrenade1.setText(killMenuTPickMenuGrenade1);

}else if(killMenuTPickMenuGrenade2 == null){

killMenuTPickMenuGrenade2 = "Smoke";

tGrenade2.setText(killMenuTPickMenuGrenade2);

}else if(killMenuTPickMenuGrenade3 == null){

killMenuTPickMenuGrenade3 = "Smoke";

tGrenade3.setText(killMenuTPickMenuGrenade3);

}else if(killMenuTPickMenuGrenade4 == null){

killMenuTPickMenuGrenade4 = "Smoke";

tGrenade4.setText(killMenuTPickMenuGrenade4);

}

bSmoke.setEnabled(false);

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

if(source == bIncendiary){

if(killMenuTPickMenuGrenade1 == null){

killMenuTPickMenuGrenade1 = "Incendiary Grenade";

tGrenade1.setText(killMenuTPickMenuGrenade1);

}else if(killMenuTPickMenuGrenade2 == null){

killMenuTPickMenuGrenade2 = "Incendiary Grenade";

tGrenade2.setText(killMenuTPickMenuGrenade2);

}else if(killMenuTPickMenuGrenade3 == null){

killMenuTPickMenuGrenade3 = "Incendiary Grenade";

tGrenade3.setText(killMenuTPickMenuGrenade3);

}else if(killMenuTPickMenuGrenade4 == null){

killMenuTPickMenuGrenade4 = "Incendiary Grenade";

tGrenade4.setText(killMenuTPickMenuGrenade4);

}

bMolotov.setEnabled(false);

bIncendiary.setEnabled(false);

if(killMenuTPickMenuGrenade1 != null && killMenuTPickMenuGrenade2 != null && killMenuTPickMenuGrenade3 != null && killMenuTPickMenuGrenade4 != null){

bMolotov.setEnabled(false);

bDecoy.setEnabled(false);

bFlashbang.setEnabled(false);

bHEGrenade.setEnabled(false);

bSmoke.setEnabled(false);

bIncendiary.setEnabled(false);

}

}

//pistols

if (source == bGlock18){

killMenuTPickMenuWeapon1 = "Glock-18";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if (source == bDualBerettas){

killMenuTPickMenuWeapon1 = "Dual Berettas";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if (source == bP250){

killMenuTPickMenuWeapon1 = "P250";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if (source == bCZ75Auto){

killMenuTPickMenuWeapon1 = "CZ75-Auto";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if(source == bDesertEagle){

killMenuTPickMenuWeapon1 = "Desert Eagle";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if(source == bR8Revolver){

killMenuTPickMenuWeapon1 = "R8 Revolver";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if(source == bTec9 ){

killMenuTPickMenuWeapon1 = "Tec-9";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if(source == bUSPS){

killMenuTPickMenuWeapon1 = "USP-S";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if(source == bP2000){

killMenuTPickMenuWeapon1 = "P2000";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

if(source == bFiveSeveN){

killMenuTPickMenuWeapon1 = "Five-SeveN";

tWeapon1.setText(killMenuTPickMenuWeapon1);

}

//zeus

if(source == bZeusx27){

killMenuTPickMenuZeus = "Zeus";

tZeus.setText(killMenuTPickMenuZeus);

}

if(source == bKits){

killMenuTPickMenuKits = "Kits";

tKits.setText(killMenuTPickMenuKits);

}

//smgs

if(source == bMAC10){

killMenuTPickMenuWeapon2 = "MAC-10";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bMP5SD){

killMenuTPickMenuWeapon2 = "MP5-SD";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bMP7){

killMenuTPickMenuWeapon2 = "MP7";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bUMP45){

killMenuTPickMenuWeapon2 = "UMP-45";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bP90){

killMenuTPickMenuWeapon2 = "P90";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bPPBizon){

killMenuTPickMenuWeapon2 = "PP-Bizon";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bMP9){

killMenuTPickMenuWeapon2 = "MP9";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

//rifles

if(source == bGalilAR){

killMenuTPickMenuWeapon2 = "Galil AR";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bAK47){

killMenuTPickMenuWeapon2 = "AK-47";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bSSG08){

killMenuTPickMenuWeapon2 = "SSG 08";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bSG553){

killMenuTPickMenuWeapon2 = "SG 553";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bAWP){

killMenuTPickMenuWeapon2 = "AWP";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bG3SG1){

killMenuTPickMenuWeapon2 = "G3SG1";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bFAMAS){

killMenuTPickMenuWeapon2 = "FAMAS";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bM4A4){

killMenuTPickMenuWeapon2 = "M4A4";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bM4A1S){

killMenuTPickMenuWeapon2 = "M4A1-S";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bAUG){

killMenuTPickMenuWeapon2 = "AUG";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bSCAR20){

killMenuTPickMenuWeapon2 = "SCAR-20";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

//heavy

if(source == bNova){

killMenuTPickMenuWeapon2 = "Nova";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bXM1014){

killMenuTPickMenuWeapon2 = "XM1014";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bSawedOff){

killMenuTPickMenuWeapon2 = "Sawed-Off";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bM249){

killMenuTPickMenuWeapon2 = "M249";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bNegev){

killMenuTPickMenuWeapon2 = "Negev";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if(source == bMAG7){

killMenuTPickMenuWeapon2 = "MAG-7";

tWeapon2.setText(killMenuTPickMenuWeapon2);

}

if (source == bQuitkillMenuTPickMenu){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ?, kits = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,killMenuTPickMenuWeapon1);

preparedStatement.setString(2,killMenuTPickMenuWeapon2);

preparedStatement.setString(3,killMenuTPickMenuGrenade1);

preparedStatement.setString(4,killMenuTPickMenuGrenade2);

preparedStatement.setString(5,killMenuTPickMenuGrenade3);

preparedStatement.setString(6,killMenuTPickMenuGrenade4);

preparedStatement.setString(7,killMenuTPickMenuZeus);

preparedStatement.setString(8,killMenuTPickMenuKits);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new killMenuTPickDrop();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class ResultMenu extends EasyApp

{

public static void main(String[] args)

{

new ResultMenu();

}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,650,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bSurvive = addButton("Did you survive?",0,120,220,100, this);

Button bRoundWin = addButton("Did you win the round?",0,220,220,100, this);

Button bBombPlant = addButton("Did your team plant the bomb?",300,120,220,100, this);

Button bTeamElimination = addButton("Win by team elimination",600,120,220,100, this);

Button bBombDetonation = addButton("Win by bomb detonation",600,220,220,100, this);

Button bBombDefusal = addButton("Win by bomb defusal",600,320,220,100, this);

Button bTimeWin = addButton("Win by time",600,420,220,100, this);

Button bLoseElimination = addButton("Lose by team elimination",850,120,220,100, this);

Button bLoseBombDetonation = addButton("Lose by bomb detonation",850,220,220,100, this);

Button bLoseBombDefusal = addButton("Lose by bomb defusal",850,320,220,100, this);

Button bLoseTime = addButton("Lose by time",850,420,220,100, this);

Button bContinueResultMenu = addButton("Continue",850,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,650,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

int intRSBalanceResultMenu;

int intSQLRoundCTResultMenu;

int intSQLRoundTResultMenu;

int intSQLRoundNumberResultMenu;

int intSQLloseBonusCount;

int intTempLoseBonusForThisRound;

String sqlSidestring;

String yourBombPlantString;

String roundWinStringCondition;

String teamBombPlantString;

String surviveString;

String winAfterPistolRoundString;

String sqlStringCTuspsORp2000;

public ResultMenu() // Constructor - change window appearance

{

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCount = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCount.next() ){

String sqlRoundCountstring = sqlRoundCount.getString(1);

intSQLRoundNumberResultMenu = Integer.parseInt (sqlRoundCountstring);}

ResultSet rsBalance = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalance.next() ){

String rs1Balancestring = rsBalance.getString(1);

intRSBalanceResultMenu = Integer.parseInt (rs1Balancestring);}

ResultSet sqlRoundCT = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCT.next() ){

String sqlRoundCTstring = sqlRoundCT.getString(1);

intSQLRoundCTResultMenu = Integer.parseInt (sqlRoundCTstring);}

ResultSet sqlRoundT = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundT.next() ){

String sqlRoundTstring = sqlRoundT.getString(1);

intSQLRoundTResultMenu = Integer.parseInt (sqlRoundTstring);}

ResultSet sqlSide = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlSide.next() ){

sqlSidestring = sqlSide.getString(1);}

ResultSet sqlYourBombPlant = stmt.executeQuery( "SELECT BombPlant FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlYourBombPlant.next() ){

yourBombPlantString = sqlYourBombPlant.getString(1);}

ResultSet sqlloseBonusCount = stmt.executeQuery( "SELECT loseBonusCount FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlloseBonusCount.next() ){

String loseBonusCountString = sqlloseBonusCount.getString(1);

intSQLloseBonusCount = Integer.parseInt(loseBonusCountString);

intTempLoseBonusForThisRound = intSQLloseBonusCount; }

ResultSet sqlwinAfterPistolRound = stmt.executeQuery( "SELECT winAfterPistolRound FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlwinAfterPistolRound.next() ){

winAfterPistolRoundString = sqlwinAfterPistolRound.getString(1);

}

ResultSet sqlCTuspsORp2000 = stmt.executeQuery( "SELECT CTuspsORp2000 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlCTuspsORp2000.next() ){ //alias

sqlStringCTuspsORp2000 = sqlCTuspsORp2000.getString(1);}

//finish it

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1920,1080);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(intSQLRoundNumberResultMenu + "");

tBalance.setText(intRSBalanceResultMenu + "$");

tCTscore.setText(intSQLRoundCTResultMenu + "");

tTscore.setText(intSQLRoundTResultMenu+ "");

bRoundWin.setEnabled(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(false);

if(yourBombPlantString.equals("True")){

teamBombPlantString = "True";

}

}

public void actions(Object source,String command){

if(source == bSurvive){

boolean SurviveBoolean = false;

while(SurviveBoolean == false){

String SurviveStringQuestion = inputString("Did you survive? Y/N");

if(SurviveStringQuestion.equals("Y")){

surviveString = "True";

SurviveBoolean = true;

}else if(SurviveStringQuestion.equals("N")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ?, buttonSurvive = ?, Armor = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,null);

preparedStatement.setString(2,null);

preparedStatement.setString(3,null);

preparedStatement.setString(4,null);

preparedStatement.setString(5,null);

preparedStatement.setString(6,null);

preparedStatement.setString(7,"True");

preparedStatement.setString(8,null);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

if(sqlSidestring.equals("TSide")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"Glock-18");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(sqlSidestring.equals("CTSide")){

if(sqlStringCTuspsORp2000.equals("CTUSPS")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"USP-S");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(sqlStringCTuspsORp2000.equals("CTP2000")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"P2000");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

}

surviveString = "False";

SurviveBoolean = true;

}else{

SurviveBoolean = false;

}

}

bRoundWin.setEnabled(true);

bSurvive.setEnabled(false);

}

if(source == bRoundWin){

boolean RoundWin = false;

while(RoundWin == false){

String RoundWinStringQuestion = inputString("Did you win the round? Y/N");

if(RoundWinStringQuestion.equals("Y")){

if(sqlSidestring.equals("TSide")){

if(yourBombPlantString.equals("False")){

bBombPlant.setVisible(true);

}

bTeamElimination.setVisible(true);

bBombDetonation.setVisible(true);

}

if(sqlSidestring.equals("CTSide")){

bTeamElimination.setVisible(true);

bBombDefusal.setVisible(true);

bTimeWin.setVisible(true);

}

if(winAfterPistolRoundString.equals("False")){

winAfterPistolRoundString = "True";

intSQLloseBonusCount = intSQLloseBonusCount - 2;

if(intSQLloseBonusCount < 0){

intSQLloseBonusCount = 0;

}

}

if(winAfterPistolRoundString.equals("True")){

intSQLloseBonusCount = intSQLloseBonusCount - 1;

if(intSQLloseBonusCount < 0){

intSQLloseBonusCount = 0;

}

}

roundWinStringCondition = "Y";

RoundWin = true;

bRoundWin.setEnabled(false);

}else if(RoundWinStringQuestion.equals("N")){

if(sqlSidestring.equals("TSide")){

if(yourBombPlantString.equals("False")){

bBombPlant.setVisible(true);

}

if(yourBombPlantString.equals("True")){

if(surviveString.equals("False")){

bLoseElimination.setVisible(true);

}

bLoseBombDefusal.setVisible(true);

}

}

if(sqlSidestring.equals("CTSide")){

if(surviveString.equals("False")){

bLoseElimination.setVisible(true);

}

bLoseBombDetonation.setVisible(true);

}

intSQLloseBonusCount = intSQLloseBonusCount + 1;

if(intSQLloseBonusCount > 4){

intSQLloseBonusCount = 4;

}

roundWinStringCondition = "N";

RoundWin = true;

bRoundWin.setEnabled(false);

}else{

RoundWin = false;

}

}

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int loseBonusUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET loseBonusCount = "+intSQLloseBonusCount+" WHERE idNumCS = 1;");

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET winAfterPistolRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, winAfterPistolRoundString);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(source == bBombPlant){

boolean BombPlantBoolean1 = false;

while(BombPlantBoolean1 == false){

String BombPlant1StringQuestion = inputString("Did your team plant the bomb? Y/N");

if(BombPlant1StringQuestion.equals("Y")){

teamBombPlantString = "True"; //alias

if(roundWinStringCondition == "Y"){

bTeamElimination.setVisible(true);

bBombDetonation.setVisible(true);

}

if(roundWinStringCondition == "N"){

bLoseBombDefusal.setVisible(true);

}

BombPlantBoolean1 = true;

}else if(BombPlant1StringQuestion.equals("N")){

teamBombPlantString = "False";

if(roundWinStringCondition == "Y"){

bTeamElimination.setVisible(true);

}

if(roundWinStringCondition == "N"){

if(surviveString.equals("False")){

bLoseElimination.setVisible(true);

}

bLoseBombDefusal.setVisible(true);

bLoseTime.setVisible(true);

}

BombPlantBoolean1 = true;

}else{

BombPlantBoolean1 = false;

}

}

bBombPlant.setEnabled(false);

}

if(source == bBombDetonation){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3500;

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

tBalance.setText(intRSBalanceResultMenu + "$");

intSQLRoundTResultMenu = intSQLRoundTResultMenu + 1;

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1; //use

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundT = "+intSQLRoundTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Win by bomb detonation");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bTeamElimination){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3250;

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

tBalance.setText(intRSBalanceResultMenu + "$");

if(sqlSidestring.equals("TSide")){

intSQLRoundTResultMenu = intSQLRoundTResultMenu + 1;

}

if(sqlSidestring.equals("CTSide")){

intSQLRoundCTResultMenu = intSQLRoundCTResultMenu + 1;

}

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundT = "+intSQLRoundTResultMenu+" WHERE idNumCS = 1;");

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+intSQLRoundCTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Win by team elimination");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bBombDefusal){

boolean BombDefusalBoolean1 = false;

while(BombDefusalBoolean1 == false){

String BombDefusal1StringQuestion = inputString("Did you defuse the bomb yourself? Y/N");

if(BombDefusal1StringQuestion.equals("Y")){

intRSBalanceResultMenu = intRSBalanceResultMenu + 300;

BombDefusalBoolean1 = true;

}else if(BombDefusal1StringQuestion.equals("N")){

BombDefusalBoolean1 = true;

}else{

BombDefusalBoolean1 = false;

}

}

intRSBalanceResultMenu = intRSBalanceResultMenu + 3500;

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

intSQLRoundCTResultMenu = intSQLRoundCTResultMenu + 1;

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

tBalance.setText(intRSBalanceResultMenu + "$");

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+intSQLRoundCTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Win by bomb defusal");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bTimeWin){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3250;

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

intSQLRoundCTResultMenu = intSQLRoundCTResultMenu + 1;

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

tBalance.setText(intRSBalanceResultMenu + "$");

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+intSQLRoundCTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Win by time");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bLoseElimination){

if(intTempLoseBonusForThisRound == 0){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1400;

}else if(intTempLoseBonusForThisRound == 1){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1900;

}else if(intTempLoseBonusForThisRound == 2){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2400;

}else if(intTempLoseBonusForThisRound == 3){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2900;

}else if(intTempLoseBonusForThisRound == 4){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3400;

}

if(teamBombPlantString.equals("True")){

intRSBalanceResultMenu = intRSBalanceResultMenu + 800;

}

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

tBalance.setText(intRSBalanceResultMenu + "$");

if(sqlSidestring.equals("TSide")){

intSQLRoundCTResultMenu = intSQLRoundCTResultMenu + 1;

}

if(sqlSidestring.equals("CTSide")){

intSQLRoundTResultMenu = intSQLRoundTResultMenu + 1;

}

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundT = "+intSQLRoundTResultMenu+" WHERE idNumCS = 1;");

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+intSQLRoundCTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Lose by team elimination");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bLoseBombDetonation){

if(intTempLoseBonusForThisRound == 0){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1400;

}else if(intTempLoseBonusForThisRound == 1){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1900;

}else if(intTempLoseBonusForThisRound == 2){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2400;

}else if(intTempLoseBonusForThisRound == 3){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2900;

}else if(intTempLoseBonusForThisRound == 4){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3400;

}

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

tBalance.setText(intRSBalanceResultMenu + "$");

intSQLRoundTResultMenu = intSQLRoundTResultMenu + 1;

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundT = "+intSQLRoundTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Lose by bomb detonation");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bLoseBombDefusal){

if(intTempLoseBonusForThisRound == 0){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1400;

}else if(intTempLoseBonusForThisRound == 1){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1900;

}else if(intTempLoseBonusForThisRound == 2){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2400;

}else if(intTempLoseBonusForThisRound == 3){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2900;

}else if(intTempLoseBonusForThisRound == 4){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3400;

}

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

tBalance.setText(intRSBalanceResultMenu + "$");

intSQLRoundCTResultMenu = intSQLRoundCTResultMenu + 1;

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+intSQLRoundCTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Lose by bomb defusal");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

if(source == bLoseTime){

//finishC

if(surviveString.equals("False")){

if(intTempLoseBonusForThisRound == 0){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1400;

}else if(intTempLoseBonusForThisRound == 1){

intRSBalanceResultMenu = intRSBalanceResultMenu + 1900;

}else if(intTempLoseBonusForThisRound == 2){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2400;

}else if(intTempLoseBonusForThisRound == 3){

intRSBalanceResultMenu = intRSBalanceResultMenu + 2900;

}else if(intTempLoseBonusForThisRound == 4){

intRSBalanceResultMenu = intRSBalanceResultMenu + 3400;

}

if (intRSBalanceResultMenu > 16000){

intRSBalanceResultMenu = 16000;

}

}else if(surviveString.equals("True")){

intRSBalanceResultMenu = intRSBalanceResultMenu + 0;

}

tBalance.setText(intRSBalanceResultMenu + "$");

intSQLRoundCTResultMenu = intSQLRoundCTResultMenu + 1;

intSQLRoundNumberResultMenu = intSQLRoundNumberResultMenu + 1;

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = "+intRSBalanceResultMenu+" WHERE idNumCS = 1;");

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+intSQLRoundCTResultMenu+" WHERE idNumCS = 1;");

int roundCountUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCount = "+intSQLRoundNumberResultMenu+" WHERE idNumCS = 1;"); //use

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET howDidYouWinTheRound = ? WHERE idNumCS = 1";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1, "Lose by time");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

bSurvive.setVisible(false);

bRoundWin.setVisible(false);

bBombPlant.setVisible(false);

bTeamElimination.setVisible(false);

bBombDetonation.setVisible(false);

bBombDefusal.setVisible(false);

bTimeWin.setVisible(false);

bLoseElimination.setVisible(false);

bLoseBombDetonation.setVisible(false);

bLoseBombDefusal.setVisible(false);

bLoseTime.setVisible(false);

bContinueResultMenu.setVisible(true);

}

//finish the rest of types of losing

if(source == bContinueResultMenu){

new ResultRound();

dispose();

}

}

}

package myProject;

import java.awt.\*;

import java.sql.\*;

public class ResultRound extends EasyApp

{

public static void main(String[] args)

{ new ResultRound();}

TextField tCTscore = addTextField("0",350,50,31,30,this);

TextField tTscore = addTextField("0",382,50,31,30,this);

TextField tRoundNum = addTextField("0",500,800,31,30,this);

TextField tBalance = addTextField("0",900,50,62,30,this);

Button bCountinue = addButton("Continue",850,550,150,100,this);

Label lCT = addLabel("CT",330,50,50,30,this);

Label lT = addLabel("T",420,50,50,30,this);

Label lRoundNum = addLabel("Round: ",455,800,50,30,this);

Label lBalance = addLabel("Balance: ",848,50,62,30,this);

Label lDropWeapon1 = addLabel("",337,100,200,30,this);

Label lDropWeapon2 = addLabel("",337,200,200,30,this);

Label lDropZeus = addLabel("",337,300,200,30,this);

Label lDropGrenade1 = addLabel("",337,400,200,30,this);

Label lDropGrenade2 = addLabel("",337,500,200,30,this);

Label lDropGrenade3 = addLabel("",337,600,200,30,this);

Label lDropGrenade4 = addLabel("",337,700,200,30,this);

Label lRoundWin = addLabel("",700,120,700,200,this);

Label lEndGame = addLabel("",500,220,400,200,this); //alias

Label lGameOver = addLabel("",500,320,400,200,this);

int RoundCountresultRound;

int BalanceresultRound;

int RoundCTresultRound;

int RoundTresultRound;

String resultRoundWeapon1;

String resultRoundWeapon2;

String resultRoundZeus;

String resultRoundGrenade1;

String resultRoundGrenade2;

String resultRoundGrenade3;

String resultRoundGrenade4;

String sqlSidestring;

String sqlCTuspsORp2000string;

String sqlHowDidYouWinTheRoundString;

public ResultRound(){

try //workon

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet sqlRoundCountresultRound = stmt.executeQuery( "SELECT RoundCount FROM IACompCSGO WHERE idNumCS = 1;" ) ; //change later

while( sqlRoundCountresultRound.next() ){

String sqlRoundCountstringresultRound = sqlRoundCountresultRound.getString(1);

RoundCountresultRound = Integer.parseInt (sqlRoundCountstringresultRound);}

ResultSet rsBalanceresultRound = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE idNumCS = 1;" ) ;

// Loop through the result set

while( rsBalanceresultRound.next() ){

String rs1BalancestringresultRound = rsBalanceresultRound.getString(1);

BalanceresultRound = Integer.parseInt (rs1BalancestringresultRound);}

ResultSet sqlRoundCTresultRound = stmt.executeQuery( "SELECT RoundCT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundCTresultRound.next() ){

String sqlRoundCTstringresultRound = sqlRoundCTresultRound.getString(1);

RoundCTresultRound = Integer.parseInt (sqlRoundCTstringresultRound);}

ResultSet sqlRoundTresultRound = stmt.executeQuery( "SELECT RoundT FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlRoundTresultRound .next() ){

String sqlRoundTstringresultRound = sqlRoundTresultRound .getString(1);

RoundTresultRound = Integer.parseInt (sqlRoundTstringresultRound);}

ResultSet sqlWeapon1resultRound = stmt.executeQuery( "SELECT Weapon1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon1resultRound .next() ){

resultRoundWeapon1 = sqlWeapon1resultRound .getString(1);

}

ResultSet sqlWeapon2resultRound = stmt.executeQuery( "SELECT Weapon2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlWeapon2resultRound .next() ){

resultRoundWeapon2 = sqlWeapon2resultRound .getString(1);

}

ResultSet sqlZeusresultRound = stmt.executeQuery( "SELECT Zeus FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlZeusresultRound .next() ){

resultRoundZeus = sqlZeusresultRound .getString(1);

}

ResultSet sqlGrenade1resultRound = stmt.executeQuery( "SELECT Grenade1 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade1resultRound .next() ){

resultRoundGrenade1 = sqlGrenade1resultRound .getString(1);

}

ResultSet sqlGrenade2resultRound = stmt.executeQuery( "SELECT Grenade2 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade2resultRound .next() ){

resultRoundGrenade2 = sqlGrenade2resultRound .getString(1);

}

ResultSet sqlGrenade3resultRound = stmt.executeQuery( "SELECT Grenade3 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade3resultRound .next() ){

resultRoundGrenade3 = sqlGrenade3resultRound .getString(1);

}

ResultSet sqlGrenade4resultRound = stmt.executeQuery( "SELECT Grenade4 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlGrenade4resultRound .next() ){

resultRoundGrenade4 = sqlGrenade4resultRound .getString(1);

}

ResultSet sqlHowDidYouWinTheRound = stmt.executeQuery( "SELECT howDidYouWinTheRound FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlHowDidYouWinTheRound.next() ){

sqlHowDidYouWinTheRoundString = sqlHowDidYouWinTheRound.getString(1);

}

ResultSet sqlSide = stmt.executeQuery( "SELECT SideChoice FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while( sqlSide.next() ){

sqlSidestring = sqlSide.getString(1);}

ResultSet sqlCTuspsORp2000 = stmt.executeQuery( "SELECT CTuspsORp2000 FROM IACompCSGO WHERE idNumCS = 1;" ) ;

while(sqlCTuspsORp2000.next() ){

sqlCTuspsORp2000string = sqlCTuspsORp2000.getString(1);}

// Close the result set, statement and the connection

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

setSize(1000,1000);

setTitle("CS:GO Economy - (c) 2021 Anton Zhukovin, Russia");

setBackground(new Color(255,255,180));

tRoundNum.setText(RoundCountresultRound + "");

tBalance.setText(BalanceresultRound + "$");

tCTscore.setText(RoundCTresultRound + "");

tTscore.setText(RoundTresultRound + "");

lDropWeapon1.setText(resultRoundWeapon1);

lDropWeapon2.setText(resultRoundWeapon2);

lDropZeus.setText(resultRoundZeus);

lDropGrenade1.setText(resultRoundGrenade1);

lDropGrenade2.setText(resultRoundGrenade2);

lDropGrenade3.setText(resultRoundGrenade3);

lDropGrenade4.setText(resultRoundGrenade4);

lRoundWin.setText(sqlHowDidYouWinTheRoundString);

}

public void actions(Object source,String command){

if(RoundCTresultRound == 16){ //alias

if(sqlSidestring.equals("CTSide")){

lEndGame.setText("You have won the game!");

lGameOver.setText("The game is over");

}else if(sqlSidestring.equals("TSide")){

lEndGame.setText("You have lost the game");

lGameOver.setText("The game is over");

}

}

if(RoundTresultRound == 16){ //alias

if(sqlSidestring.equals("TSide")){

lEndGame.setText("You have won the game!");

lGameOver.setText("The game is over");

}else if(sqlSidestring.equals("CTSide")){

lEndGame.setText("You have lost the game");

lGameOver.setText("The game is over");

}

}

if(RoundTresultRound == 15 && RoundCTresultRound == 15){ //alias

lEndGame.setText("It's a draw!");

lGameOver.setText("The game is over");

}

if(RoundCountresultRound == 16){ //if it is the 16th round, the sides should be changed

if(sqlSidestring.equals("TSide")){ //if the player is playing for T side, it should change to CT

sqlSidestring = "CTSide";

}else if(sqlSidestring.equals("CTSide")){ //if the player is playing for CT side, it should change to T

sqlSidestring = "TSide";

}

RoundCTresultRound = RoundTresultRound; //the score is connected to the team, not to the side, so the team should keep it score when changing side

RoundTresultRound = RoundCTresultRound; //the score is connected to the team, not to the side, so the team should keep it score when changing side

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

int balanceUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET Balance = 800 WHERE idNumCS = 1;"); //the balance should reset to original value

int roundTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundT = "+RoundTresultRound+" WHERE idNumCS = 1;"); //the score is connected to the team, not to the side, so the team should keep it score when changing side

int roundCTUpdate = stmt1.executeUpdate("UPDATE `CSGOSchema`.`IACompCSGO` SET RoundCT = "+RoundCTresultRound+" WHERE idNumCS = 1;"); //the score is connected to the team, not to the side, so the team should keep it score when changing side

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ?, Weapon2 = ?, Grenade1 = ?, Grenade2 = ?, Grenade3 =?, Grenade4 =?, Zeus = ?, Armor = ?, SideChoice = ? WHERE idNumCS = 1;"; //everything should be set to default when changing side

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,null);

preparedStatement.setString(2,null);

preparedStatement.setString(3,null);

preparedStatement.setString(4,null);

preparedStatement.setString(5,null);

preparedStatement.setString(6,null);

preparedStatement.setString(7,null);

preparedStatement.setString(8,null);

preparedStatement.setString(9,sqlSidestring);

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

if(sqlSidestring.equals("TSide")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"Glock-18");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

if(sqlSidestring.equals("CTSide")){

if(sqlCTuspsORp2000string.equals("CTUSPS")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"USP-S");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}else if(sqlCTuspsORp2000string.equals("CTP2000")){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET Weapon1 = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"P2000");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

}

}

if (source == bCountinue){

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt1 = conn.createStatement() ;

Statement stmt2 = conn.createStatement() ;

// Execute the query

String query = "UPDATE `CSGOSchema`.`IACompCSGO` SET MainGameCondition = ?, HMKCondition = ?, BombPlant = ?, buttonBombPlant = ?, buttonSuicideReward = ?, buttonGrenadeUse = ?, buttonTeamKill = ?, buttonSurvive = ? WHERE idNumCS = 1;";

PreparedStatement preparedStatement = conn.prepareStatement(query);

preparedStatement.setString(1,"BuyMenu");

preparedStatement.setString(2,"False");

preparedStatement.setString(3,"False");

preparedStatement.setString(4,"False");

preparedStatement.setString(5,"False");

preparedStatement.setString(6,"False");

preparedStatement.setString(7,"False");

preparedStatement.setString(8,"False");

preparedStatement.executeUpdate();

// Loop through the result set

// Close the result set, statement and the connection

stmt1.close() ;

conn.close();

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

new MainGameT();

dispose();

if(RoundCTresultRound == 16 || RoundCTresultRound == 16 || RoundCountresultRound == 30){

dispose();

}

}

}

}

package myProject;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* \*/

/\* FILE: MysqlTest.java \*/

/\* \*/

/\* A simple JDBC example for BlueJ and MySQL \*/

/\* ========================================= \*/

/\* \*/

/\* V1.00 24-FEB-2008 Te https://www.heimetli.ch/ \*/

/\* \*/

/\* ------------------------------------------------------------------------- \*/

/\* \*/

/\* This example was coded and tested with JDK1.6.0\_01-b06 on Windows \*/

/\* and MySQL 5.0.45 running on a linux machine \*/

/\* \*/

/\* To run this example, you need a database with the following properties: \*/

/\* => user "zhaw" \*/

/\* => password "password" \*/

/\* => a database called "demo" \*/

/\* => and a table called "Cust" having at least one column \*/

/\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.sql.\* ;

/\*\*

\* This class is used to show how to access a MySQL database from code running in BlueJ

\*

\* Preparation: download Connector/J from <a href="http://dev.mysql.com/downloads/connector/j/5.1.html>dev.mysql.com</a>.

\* You can download it without registration if you look for and click the "no thanks" links ...

\*

\* You don't need the complete .ZIP, unpack just the .jar and put it into a

\* directory where ist is easy to find.

\*

\* Then start BlueJ and choose "Tools"/"Preferences". Select the "Libraries"-Tab

\* and press the "Add"-Button. Navigate to the .jar, select it and click "Open".

\* The library will appear in the table "User libraries".

\* You will probably get a warning, that the library will not be loaded until

\* the next compile. This is not a problem, therefore you can acknowledge the

\* warning.

\*

\* If necessary, customize the JDBC URL in the call to getConnection. It is documented on

\* <a href="http://dev.mysql.com/doc/refman/5.0/en/connector-j-reference-configuration-properties.html">

\* this page</a>

\*/

class MysqlTest

{

/\*\*

\* Connect to MySQL and read the table "Cust", then print the contents of the first column.

\*/

public void test()

{

try

{

// Load the database driver

Class.forName( "com.mysql.jdbc.Driver" ) ;

// Get a connection to the database

Connection conn = DriverManager.getConnection( "jdbc:mysql://127.0.0.1:3306/CSGOSchema?user=root&password=administrator" ) ;

// Print all warnings

for( SQLWarning warn = conn.getWarnings(); warn != null; warn = warn.getNextWarning() )

{

System.out.println( "SQL Warning:" ) ;

System.out.println( "State : " + warn.getSQLState() ) ;

System.out.println( "Message: " + warn.getMessage() ) ;

System.out.println( "Error : " + warn.getErrorCode() ) ;

}

// Get a statement from the connection

Statement stmt = conn.createStatement() ;

// Execute the query

ResultSet rs = stmt.executeQuery( "SELECT Balance, RoundCount FROM IACompCSGO" ) ;

ResultSet rs1 = stmt.executeQuery( "SELECT Balance FROM IACompCSGO WHERE RoundCount = 0;" ) ;

// Loop through the result set

while( rs1.next() ){

System.out.println( rs1.getString(1) );}

// Close the result set, statement and the connection

rs.close() ;

rs1.close() ;

stmt.close() ;

conn.close() ;

}

catch( SQLException se )

{

System.out.println( "SQL Exception:" ) ;

// Loop through the SQL Exceptions

while( se != null )

{

System.out.println( "State : " + se.getSQLState() ) ;

System.out.println( "Message: " + se.getMessage() ) ;

System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}

catch( Exception e )

{

System.out.println( e ) ;

}

}

}