# Virtual Chemistry Lab

## Summer Internship 2012

Submitted in fulfilment of internship project

**By**

**Development on EkShiksha Initiative**

Under the Guidance of

**Prof. D. B. Phatak**

**Avinash Awate**

**Department of Computer Science and Engineering,**

**Indian Institute of Technology, Bombay**

**Mumbai**

# CONTENTS

# 1. INTRODUCTION

## 1.1 PURPOSE

The purpose of this document is to outline the technical design of the Virtual Chemistry Lab and to provide the user specifications for the Virtual Chemistry Lab interactive simulation.
Its main purpose is to-
· provide the link between the User Specifications and the detailed Technical design documents
· to detail the functionality which will be provided by each module or group of modules and show how the various modules interact in the design

## 1.2 **DOCUMENT ORGANIZATION**

· Introduction
· Design overview
· User Requirements
· System Specifications
· Use Case Diagrams

## 1.3 **OBJECTIVE**

No government school (and many a school in the private sector) has a chemistry lab. This could be due to variety of reasons viz: because of safety, expense of procuring chemicals, lack of knowledge amongst teachers, etc. The idea in this project, is to develop a JAVA interactive simulator that will demonstrate chemistry experiments, allow students to perform them, and support evaluation. “Bring the lab to your PC ” is the motto.

## 1.4 **SCOPE**

The aim is to illustrate the concepts of chemistry through a Virtual Chemistry Lab(VCL).The VCL will allow students to carry out high school level experiments by utilization of standard equipment such as beakers, burettes, flasks and pipettes, and common chemical reagents like acids and bases. We would be using Java Applet graphics to depict the equipment and student actions in a realistic manner. The VCL would allow teachers to set up demonstration experiments from which the students can observe and learn. Finally all student actions will be automatically captured and the educators will be able to replay the experiments and evaluate the student.

# 2. DESIGN OVERVIEW

· We will be using in memory JAVA objects. The in memory objects are essentially a collection of objects and and actions performed on them. The objects would be the standard set of equipment needed in a chemistry experiment.

These objects may contain a compound or a mixture of compounds. Each compound will have a display name, its chemical formula and the amount and strength present in the mixture.

· Each object will support actions or methods commonly performed in a chemistry laboratory; for instance move,pour,wash or attach one object to another.

· We will be using an XML reader to read the experiment file and load the contents into the in memory JAVA objects.

· The play module will interpret the JAVA objects and call the respective functions to display the experiment. It will support an explanation board so that students can pause,rewind and continue to learn the experimental procedures.

· The next step would be to Create the Demonstration from the teacher's end. The initially empty chemistry workbench is supposed to be setup using chemicals from the shelf. The shelf itself has to be filled from the setup portion of the XML file.

· The teacher then carries out the experiment using the commands of move,pour,washetc and saves the demonstration.

· The teacher should annotate his or her actions so that those explanations can go into a separate database for manual translation and reused in multiple local languages.

· The next module would be for the teacher to Create the Exercise Experiment. Only the pre experiment portion of the XML file will be created.

· The teacher must create the problem description next.

· The third module would be Performing the Experiment. The student after loading the experiment file will use the play module to perform the experiment. The play module will allow the student to undo options and correct mistakes. All actions performed by the student will be saved in an XML file.

· The final module would be consisting of Evaluation and Grading of the experiment performed by the student.

# 3. USER REQUIREMENTS

This part of the document explains the various semantics and general framework of the Virtual Chemistry Laboratory and the areas on which this product is meant to serve the target audience. This must be explained from the perspectives of the two entities deeply involved in the execution of the VCL i.e. the teacher and the student.

## 3.1 **TEACHER'S SIDE METHODOLOGY**

### 3.1.1 **Demonstrate Experiment**

The first step from the teacher's side is to virtually pick up bottles full of chemicals or reagents from a store and place them on the shelf or table which form part of the workbench. The teacher can then pour the chemicals from the standard set of bottles into the various equipment like flasks,beakers,pipettes and others and carry out the experiment using general physical action such as moving the equipment around the workbench,pouring a reagent into another chemical stored in a burette,or washing them when needed to be reused. The demonstration must be retained in the teacher's memory.

### 3.1.2 **Setup Laboratory Experiment**

This step would consist of the teacher using the stores to put various chemicals on the shelf. He must also put the equipment needed for the particular experiment on the workbench. The teacher should then create the problem description for the student to guide the student to perform the experiment.

### 3.1.3 **Grade Student's Experiment**

The teacher is expected to evaluate and perform a rational grading for the student performed experiment after he has submitted his deduced results.

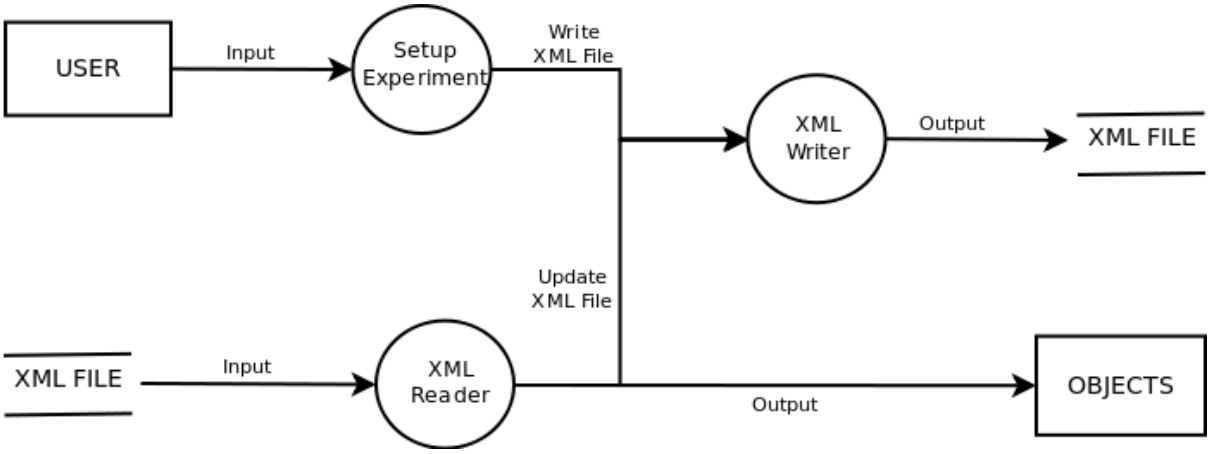
## 3.2 **STUDENT'S SIDE METHODOLOGY**

### 3.2.1 **View Demonstrations**

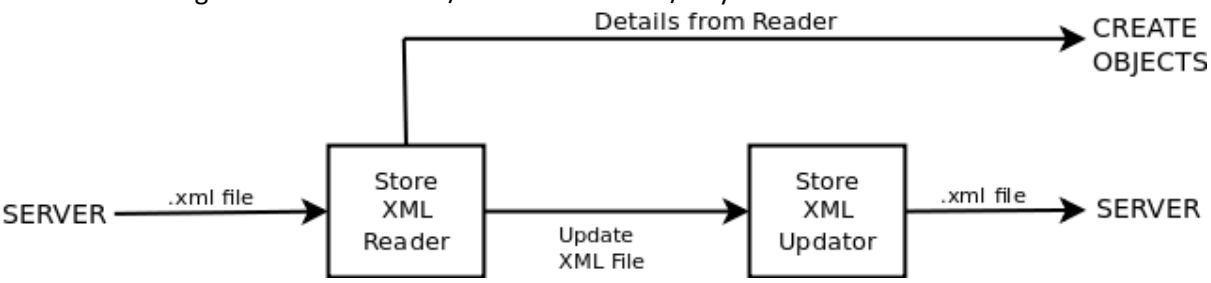
The student must be allowed to view the experiment the teacher has conducted as and whenever required. The student is allowed to pause, rewind the sequence in a limited manner and continue. The physical methods such as pour would be causing a change in the quantity of the mixture in both the source and destination equipment. The Virtual Chemistry Laboratory intends to depict these actions using animation as realistically as possible.

### **3.2.2 Conduct Experiments**

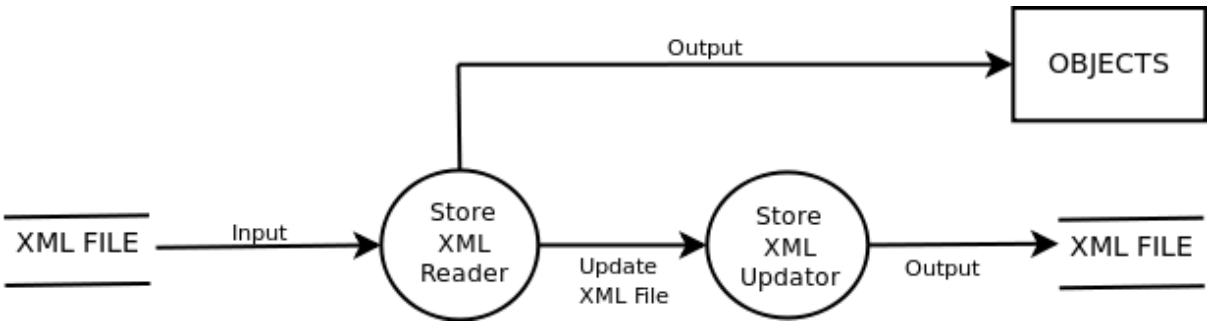
The student must be able to view the display of the chemicals on the shelf and the equipment on the workbench. The student will then start performing the experiment and he or she will be allowed to correct his mistakes if he finds any, and continue. The actions of the student must be saved for submission to the teacher so that he or she can be evaluated.




Above: Flow Diagram for XML Reader/Writer in Perform/Play Module




Above: Flow Diagram for XML Reader/Writer in Store Module




Above: Flow Diagram for XML Updater in Store Module

# 4. SYSTEM SPECIFICATIONS

**4.1 Experiment File**

Candidate: **Aviral Nigam** College: **National Institute of Technology Calicut**

The candidate's module work consists of implementation of the XML reader and writer. The XML reader is a reusable component of the system. The XML reader is supposed to read the experiment file and load the contents into the in-memory JAVA objects. Upon completion of the student's experiment the XML writer will read the data from the JAVA objects in memory and hence create an XML file which can be read by the student and played on the display. Also once the student completes his experiment it must be saved in an XML file so that the teacher can play it back on for evaluation.

The candidate in this module will also work on the structural design of the equipment of TESTTUBE.

**4.2 Handle Display**

Candidate: **MayurRao**College: **Visvesvaraya National Institute of Technology, Nagpur**

This module handles the the display features of the Virtual Chemistry Laboratory. It is concerned with drawing the workbench and displaying standard equipments such as flasks,bottles,burettes and others appropriately. It will handle the major interaction amongst elements.The issues involving resizing of the JAVA applet and its specific components to the correct proportion will be taken care of. Also there would be an interactive blackboard for the student and teacher for the purpose of displaying observations, steps , and instructions meant for the sequential completion of the experiment.

The candidate in this module will also work on the structural design of the equipment of PIPETTE.

**4.3 Play Module**

Candidate: **KaushikRaju**College: **Visvesvaraya National Institute of Technology, Nagpur**

The Play module is intended to deal with the actions and movements of the standard equipment in the chemistry laboratory in the demonstration mode. It will allow the student to view the teacher's demonstration or the teacher to view a previously done experiment by the student. It can be used to undo the previous operations, in a stepwise manner but not in a timewise sequence. The student will be allowed to play,pause and rewind and continue.

The candidate in this module will also work on the structural design of the equipment of BEAKER.

**4.4 Setup Experiment**

Candidate: **Harsh Jhamtani** College: **IIT Roorkee**

The purpose of the experiment set-up module is to allow the 'teacher' user to set up the experiment by moving the required chemicals and equipments from store to workbench, and provide various

details like the title of the experiment, description of the experiment, etc. There will also be provisions for adding a chemical from list and to add a chemical to list. On releasing the chemical bottle/equipment in the workbench, it is automatically placed in a suitable position on shelf or table. In case of chemical bottle, the user is also asked the concentration of the chemical bottle he wants to provide. In case of equipments, the user is asked to choose one of the various sizes available for that particular equipment.

**4.5 Perform Actions**

Candidate: **Anirban Mukherjee** College: **BITS, Pilani**

The candidate's work in this module would comprise implementation of the and Mouse Listener and MouseMotionListener methods to capture mouse events such as button presses, drags, releases and clicks to the JAVA application to perform move, pour, wash, burn and fill the equipment objects that would otherwise be inaccessible. The listeners will produce specific user prompts on screen to take inputs of volume to be transferred or warnings to be given in case of overflow or garbage values. All activities are subsequently recorded in the Activities list.

The candidate in this module will also work on the structural design of the equipment of BOTTLE.
.

**4.6 Equipment Design**

Candidate: **VikramVerma**College: **Maulana Azad National Institute of Technology, Bhopal**

Candidate: **PankajJangir**College: **Maulana Azad National Institute of Technology, Bhopal**

Candidate: **MalleswariVinukonda**College: **Kakinada Instiute of Engineering and Technology, Korangi**

The candidates in this module will work on the structural design of the various equipments of BURNER,BURETTE and FLASK respectively. They will be concerned with the parameters of type,size, and identification of the equipment JAVA objects and the quantity and strength of solutions in them. They must also coordinate their equipment design in accordance with the desired activities of washing, pouring, moving and others to perform the experiment in accordance with what is expected from the user requirements. They will also make the Class Diagrams.

EXPERIMENT SET-UP MODULE

The purpose of the experiment set-up module is to allow the 'teacher' user to set up the experiment by moving the required chemicals and equipments from store to workbench, and provide various details like the title of the experiment, description of the experiment, etc.

**KEY FEATURES**:

- **Reading Chemical List**
  XML reader is used for reading a list of chemicals stored as xml file.

- **Providing chemical bottles on cupboard**

Initially ten chemical bottles are provided in the cupboard.

- **Providing the various available equipments on cupboard**
  Each type of available equipment is provided in the cupboard.

- **Asking for Header Details**
  The user is prompted to enter the following 'header details' for the experiment he wants to set-up: title, author name, level, marks, description, student name, instructions.

- **Setup using 'Drag and Drop'**
  The user can transfer the required equipments and/or chemical bottles from cupboard to the workbench. For this, 'drag and drop' methodology is to be used i.e. the user is supposed to drag the required equipment/chemical from the cupboard to the workbench.

  On releasing the chemical bottle/equipment in the workbench, it is automatically placed in a suitable position on shelf or table. In case of chemical bottle, the user is also asked the concentration of the chemical bottle he wants to provide. In case of equipments, the user is asked to choose one of the various sizes available for that particular equipment.

- **File Menu Features**:

- **Add chemical from list**

Adding a chemical from the chemical list shown as a drop-down list.

- **Adding a new chemical**

To add a new chemical which is not present in the list, click the 'Add new chemical' option. The user is prompted to enter the name, formula and colour of the chemical(as RGB). Moreover, the chemical so added is appended to the initial chemical list through aXMLWriter.

- **Save the current set-up file**

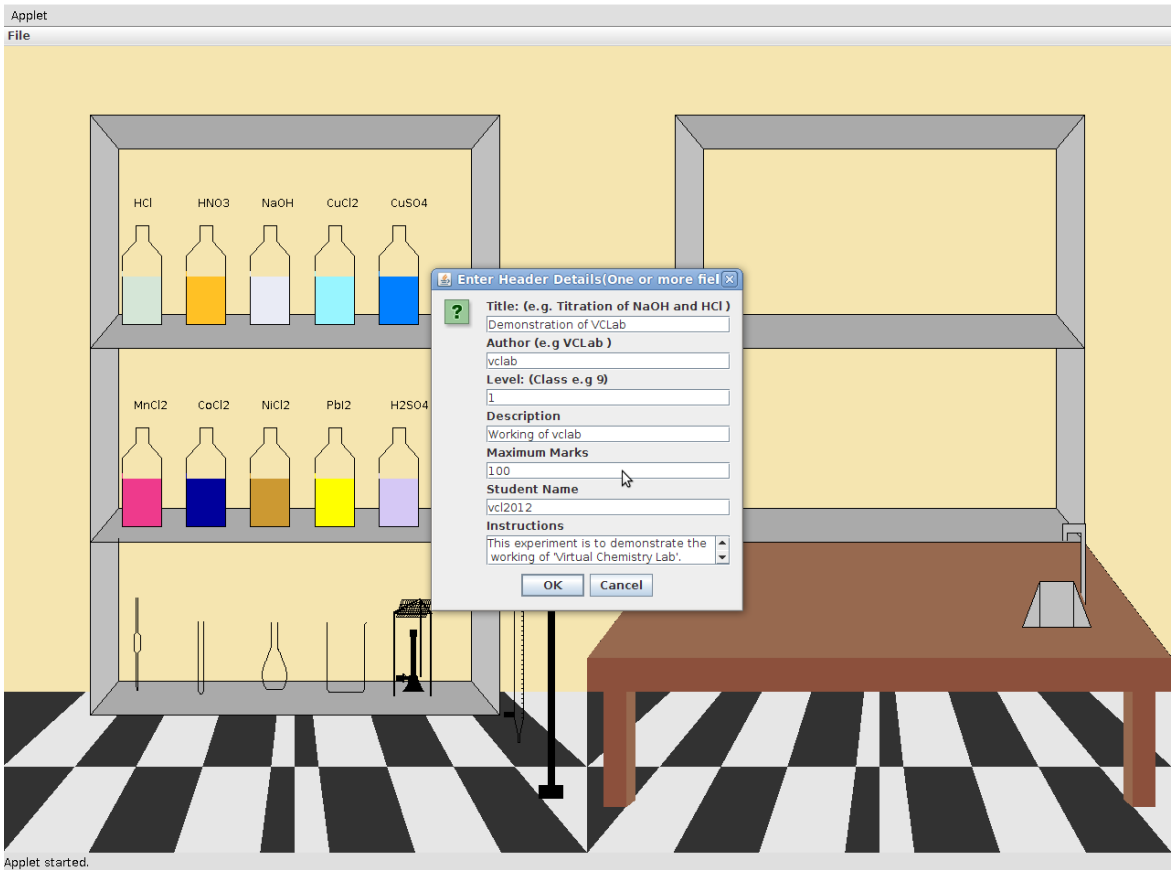
  The user can save the current experiment set-up file after adding all the required chemicals and equipments. The set-up file so saved can be opened in the 'perform' mode to start the experiment.

- **Open a pre-existing set-up file**

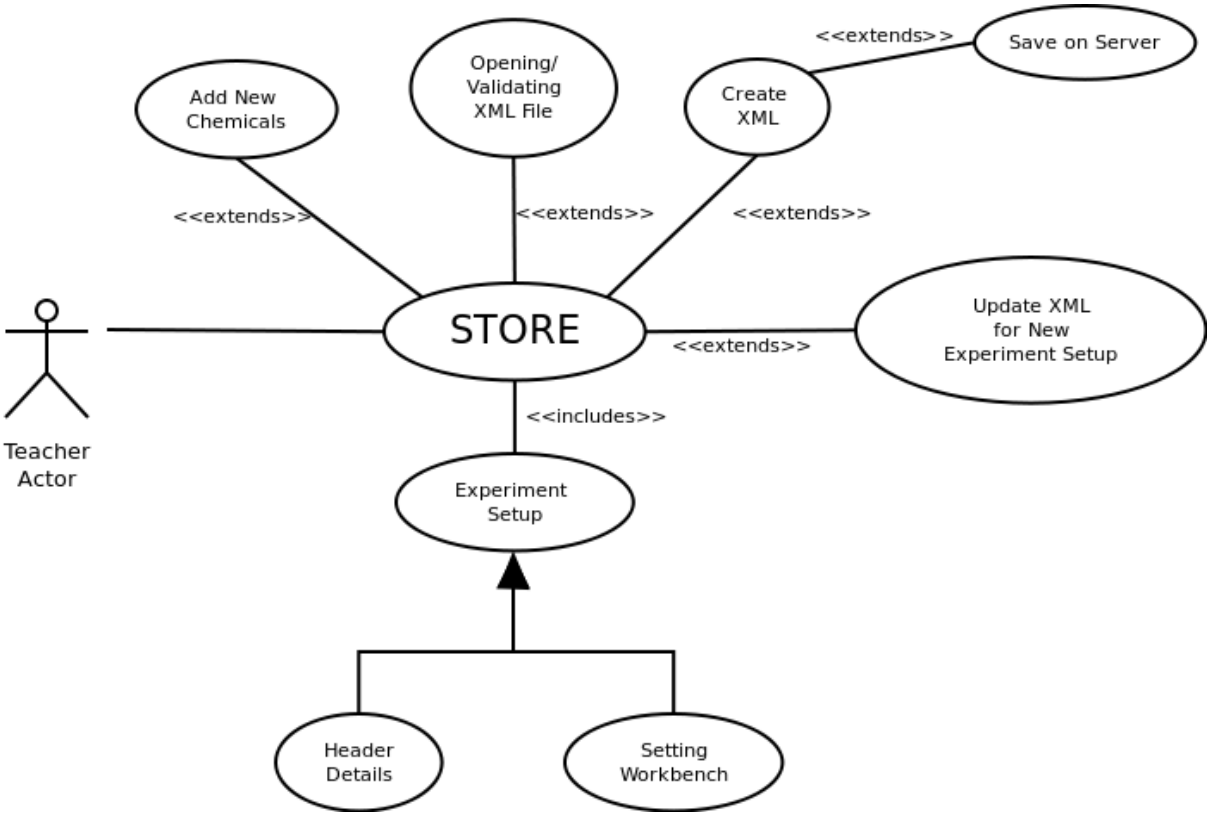
It allows user to continue the set-up from some pre-existing set-up file. The user can make the required changes and the save the new experiment set-up file.

- **Switch to perform mode**

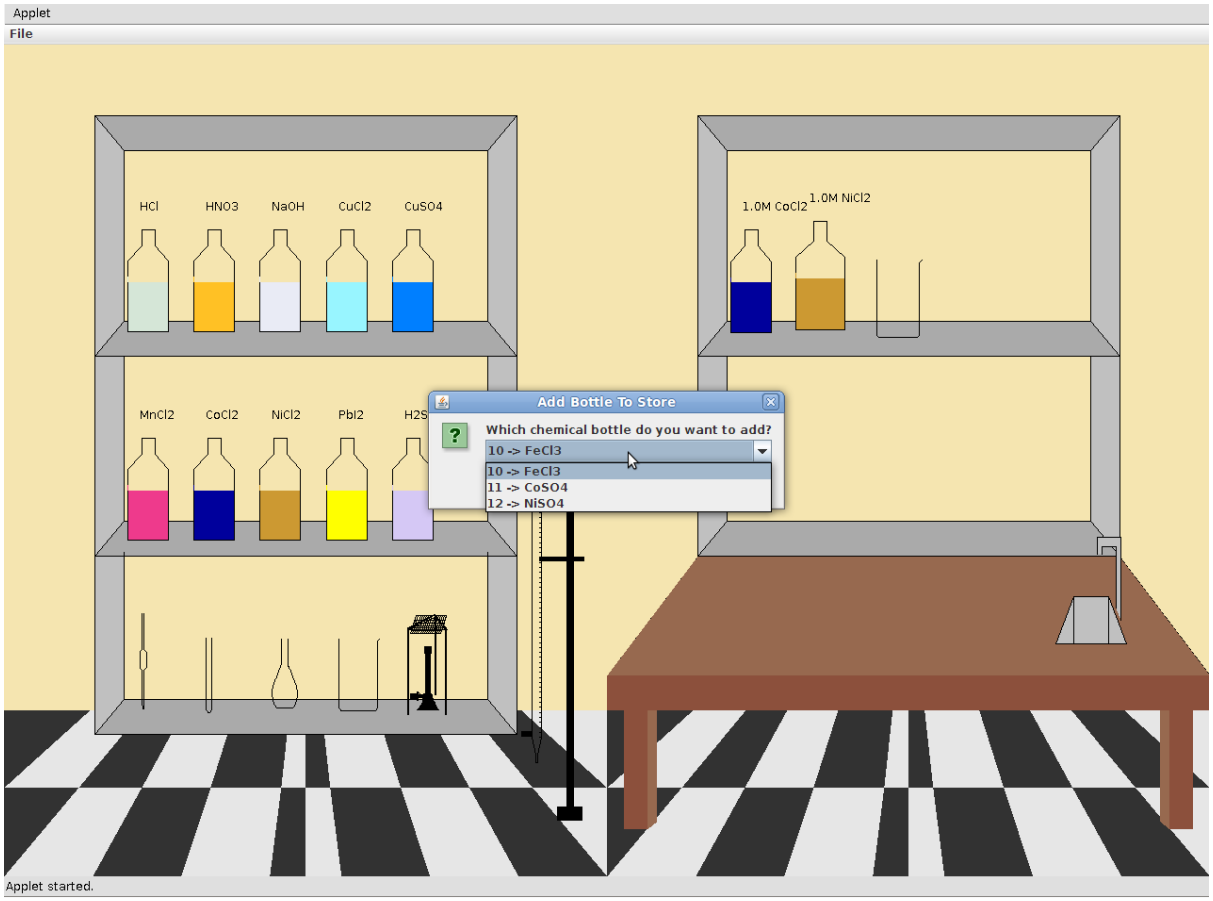
The user can click 'Perform mode' option available under the 'File' menu to switch to perform mode where he can perform various experiments.



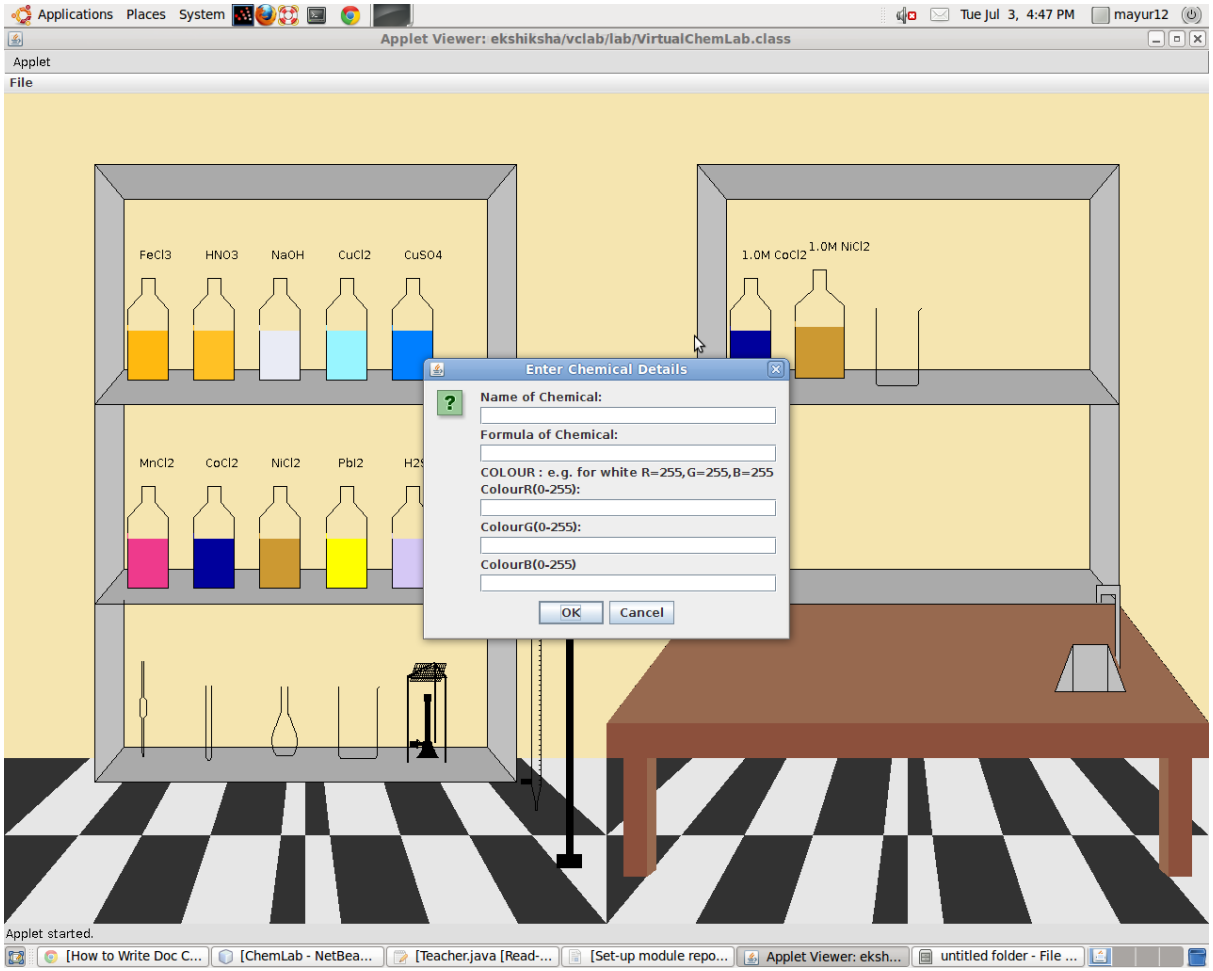

Screenshot 1.1



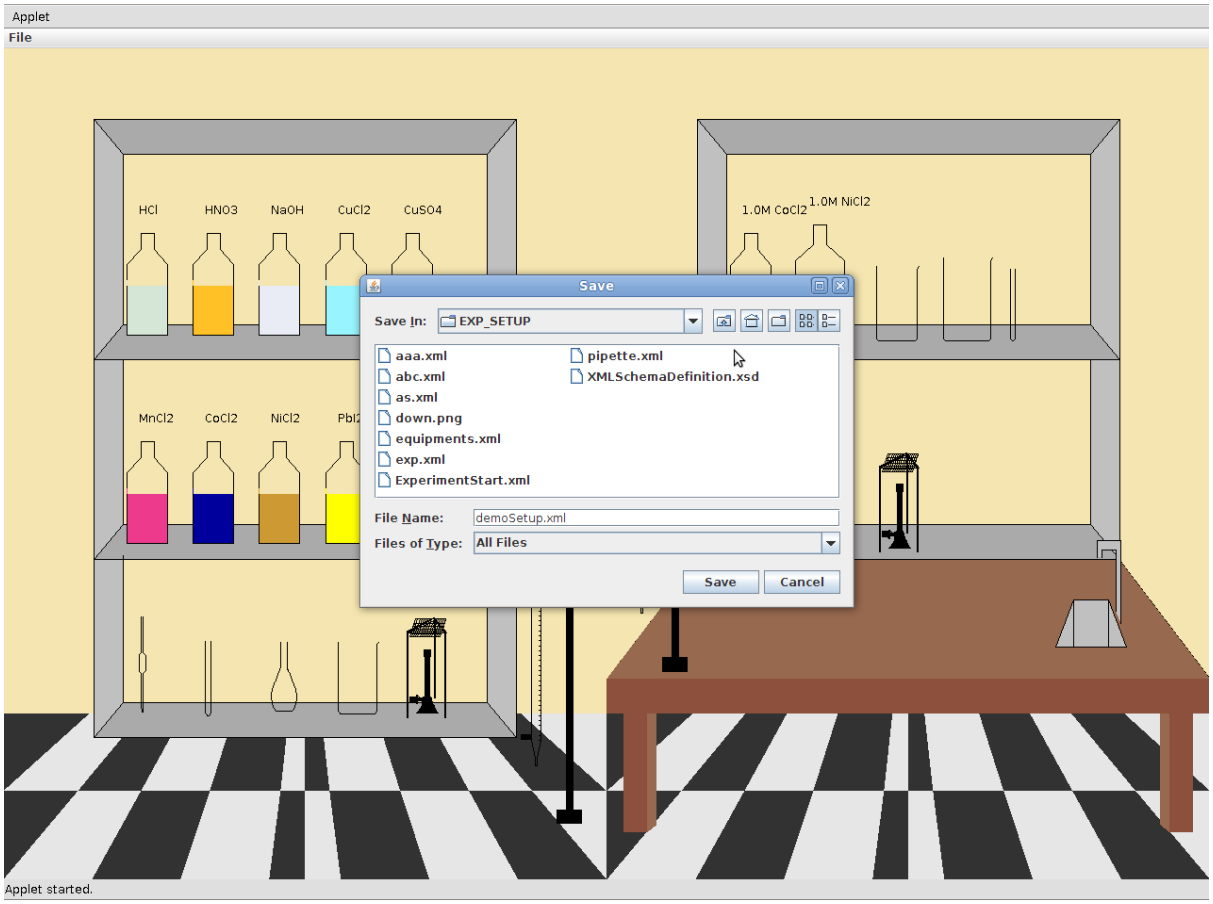

Use Case Diagram of Store Module




Screenshot 1.2




Screenshot 1.3




Screenshot 1.4

## EXPERIMENT PERFORM MODULE

The perform module is concerned with allowing the Student or Teacher entity to go ahead doing the stepwise procedure necessary to complete a sample Chemistry experiment. The WorkBench class and its Activities and Equipments Lists record all that is done by the individual and saves it,so it can be played or viewed again for future use.It is the centre of activity of the Virtual Chemistry Lab.

### Key Features

Allowing the individual to drag,drop and rotate various equipments across the workbench

- For the purpose of moving or pouring into other equipments

Allowing the individual to pour chemicals from one equipment to another with constraints

- Constraints being kept for negative or illegal values,or for quantities not present or that which will cause an overflow

Allowing the individual to wash or heat the equipments

-the individual must move it to the Basin region or coincide with the Bunsen Burner to perform these actions

Taking user input regarding how much quantity is to be withdrawn from or poured into particular equipments

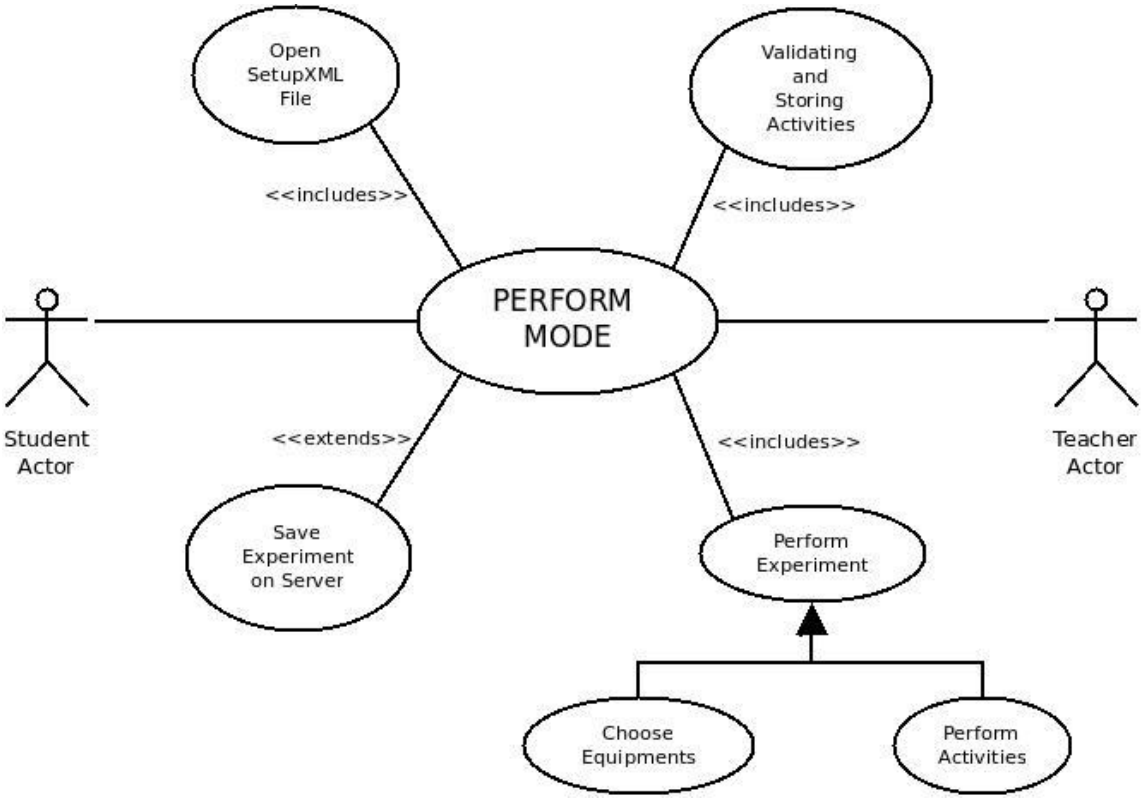
-Done with the help of a Jframe which is set Visible on and off at the necessary timings

Reusing a previous experimental set up which has been created by the Store module or by an independent xml file

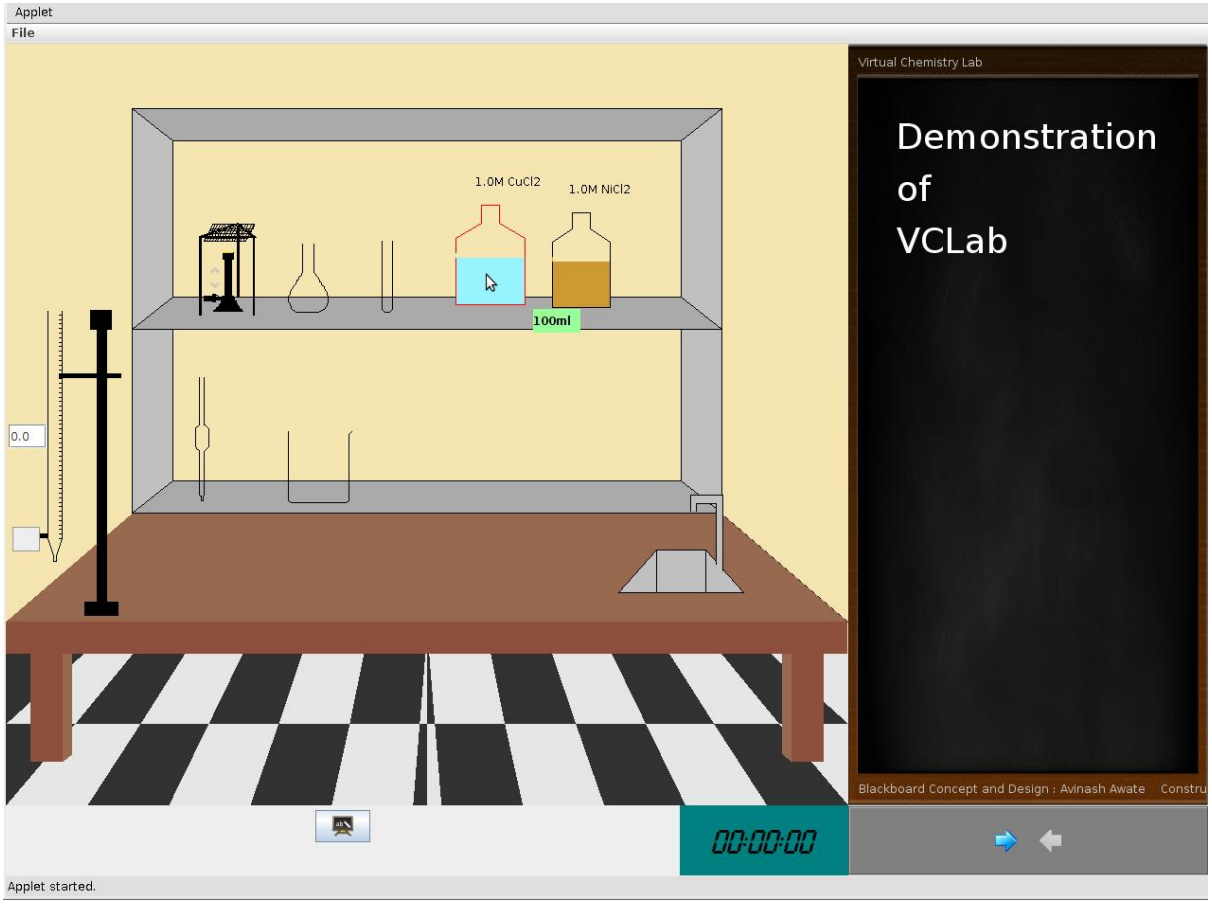
-This would help a student to re-access a previously created experimental setup for multiple time use.

Recording of all activities in the Activities arrayList to replay it in the future using the Demonstration module

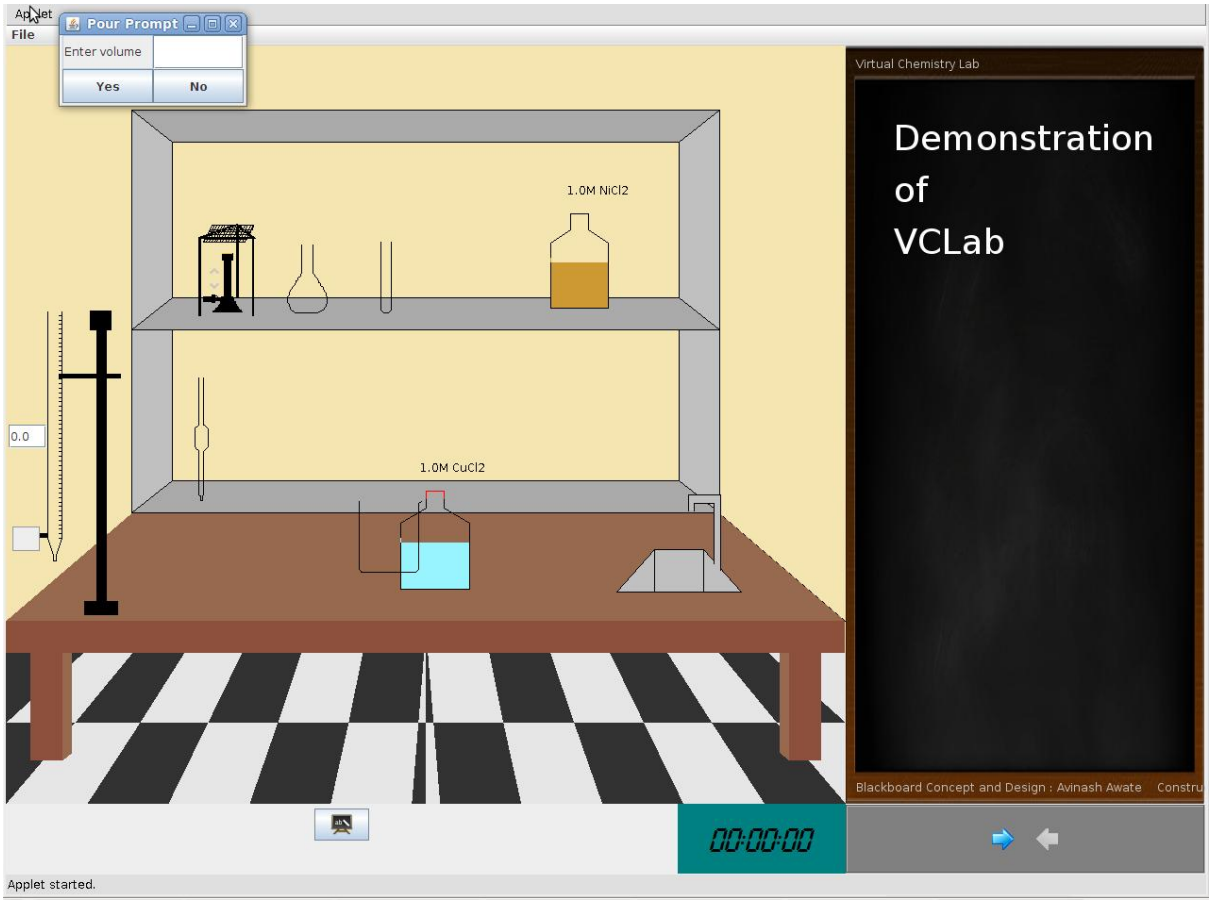
-To help the teacher replay the student's experiment for evaluation.



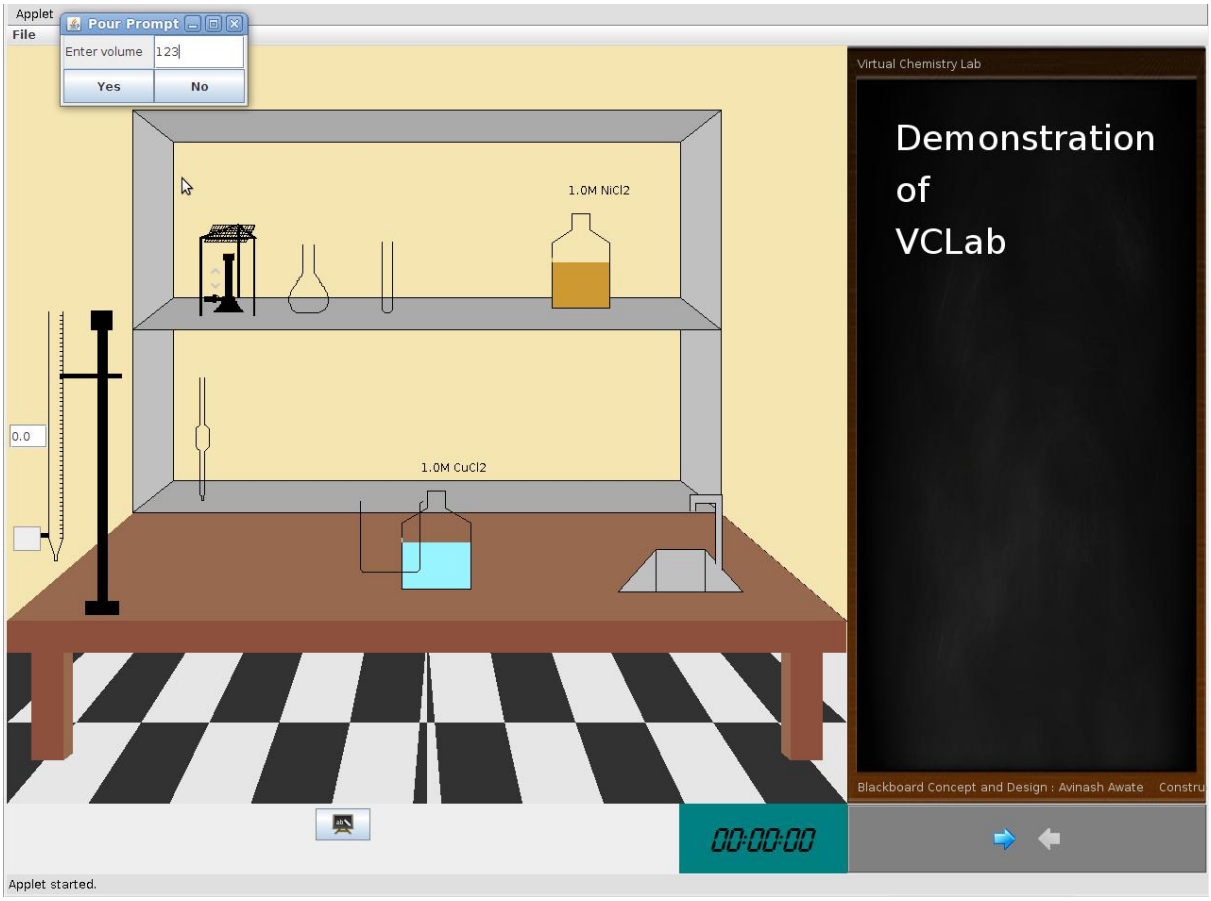

Use Case Diagram of Perform Module



Screenshot 2.1




Screenshot 2.2




Screenshot 2.3

**EXPERIMENT DEMONSTRATION MODULE**

This mode basically deals with performing the actions already stored in an XML file. Students can use this module to view an experiment performed by the teacher or otherwise. Teachers can use it as a tool to review the actions of the student-performed experiment.

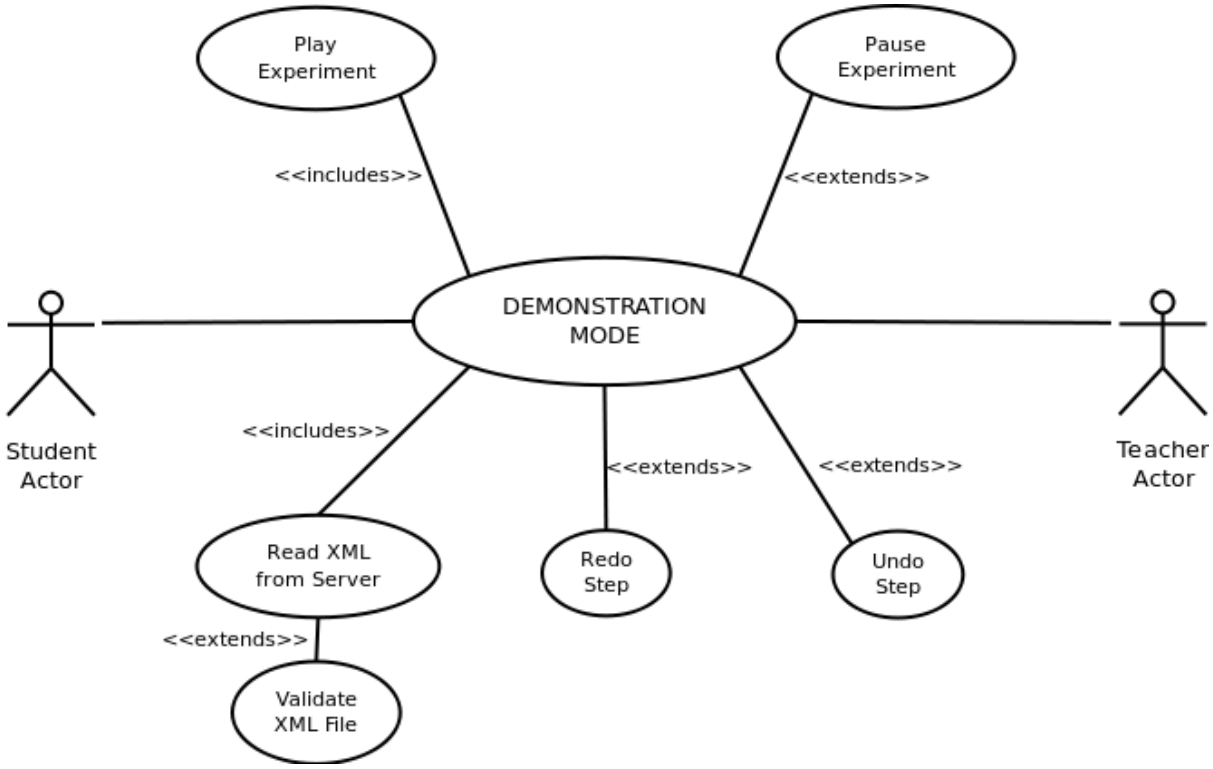
**Key features:**

**Reviewing an experiment :** Any performed experiment(saved) can be reopened to view all the activities of the same. This will help in learning a new experiment (for students) or evaluation of a performed experiment (for teachers).

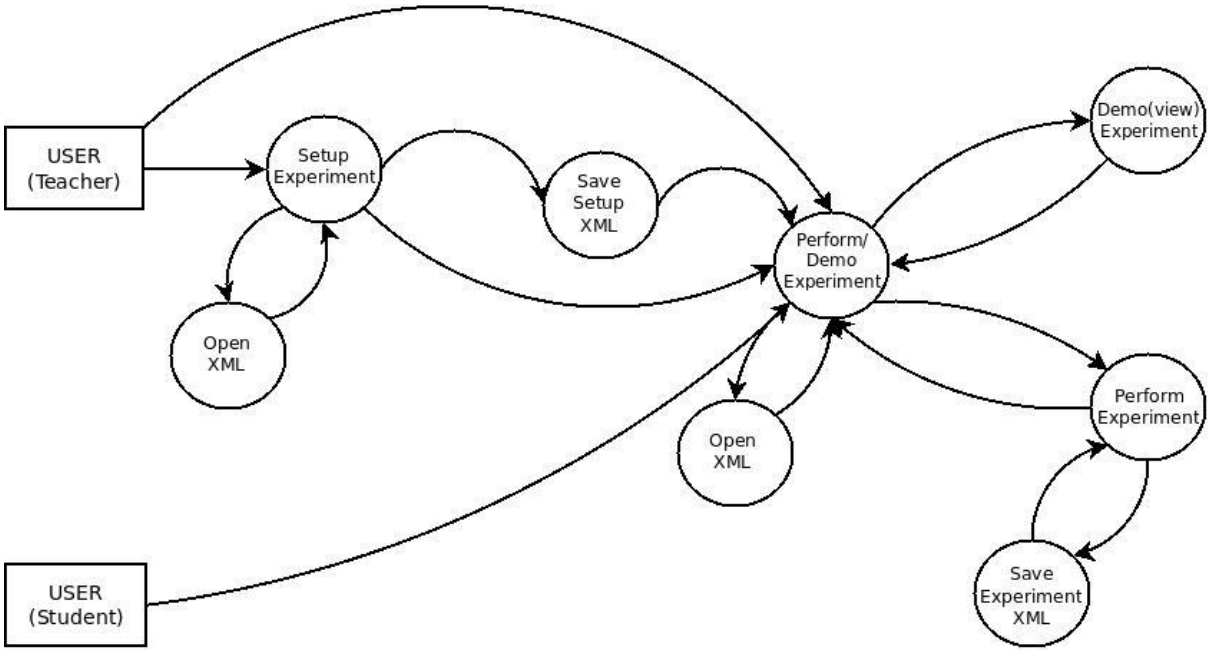
**Control of the experiment :** The user has been given independence to control the flow of the experiment. The saved experiment can be played, paused, forward-stepped, backward-stepped according to the user's needs.

**Monitoring the experiment :** The experiment is carried out in a step-wise manner, the quantities of each equipment can be monitored using the mouse pointer at each step.

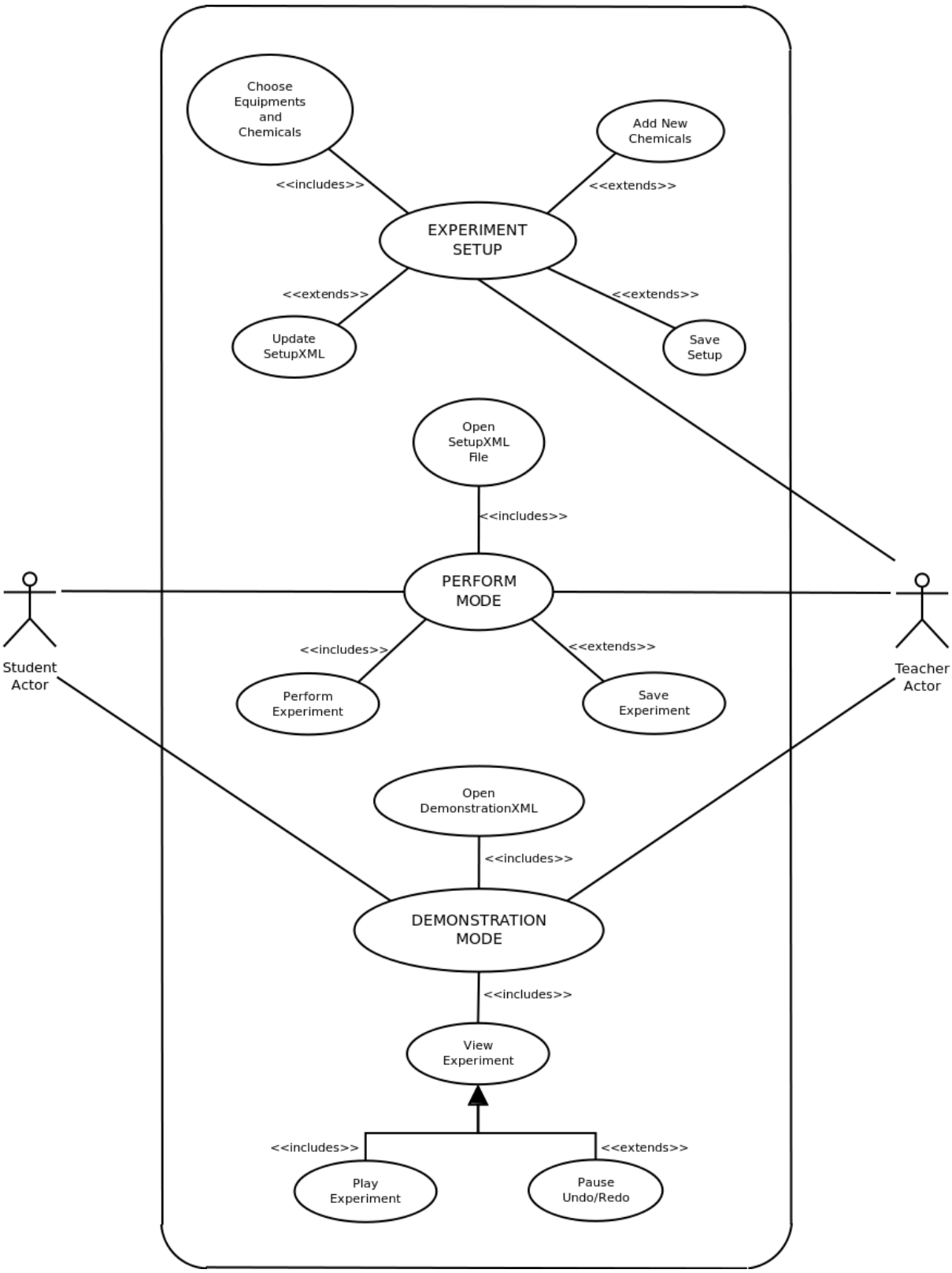
**Explanations :** A blackboard is provided which supports the explanations provided by the teacher.




Use Case Diagram of Demonstration Module




Data flow diagram


Choose Equipments and Chemicals
Add New Chemicals
<<includes>>
<<extends>>
EXPERIMENT SETUP
<<extends>>
<<extends>>
Update SetupXML
Save Setup
Open SetupXML File
<<includes>>
PERFORM MODE
Student Actor
Teacher Actor
<<includes>>
<<extends>>
Perform Experiment
Save Experiment
Open DemonstrationXML
<<includes>>
DEMONSTRATION MODE
<<includes>>
View Experiment
<<includes>>
<<extends>>
Play Experiment
Pause Undo/Redo

# 8. Package,Class and Method Details

# ekshiksha.vclab.activity

## 1.1 Class Activities

```
java.lang.Object
└ekshiksha.vclab.activity.Activities
```

---

**Constructor Detail**

Activities
```
public Activities()
```

**Method Detail**

add
```
public static void add(Activity activity)
```

---

remove
```
public static void remove(Activity activity)
```

---

getActivities
```
public static java.util.ArrayList<Activity>getActivities()
```

---

setActivities
```
public static void
setActivities(java.util.ArrayList<Activity> activityList)
```

---

removeAll
```
public static void removeAll()
```

---

1.2 Class Activity
```
java.lang.Object
└ekshiksha.vclab.activity.Activity
```
All Implemented Interfaces:
ActivityType
Direct Known Subclasses:
Burn, Label, Move, Pour, Wash, Weigh

**Method Detail**

getContentColor
```
publicjava.awt.ColorgetContentColor()
```

---

setContentColor
```
public void setContentColor(java.awt.Color contentColor)
```

---

getContent
```
publicjava.lang.StringgetContent()
```

---

setContent

```
public void setContent(java.lang.String content)
```

---

getBurnerId
```
publicintgetBurnerId()
```

---

setBurnerId
```
public void setBurnerId(int burnerId)
```

---

getContents
```
publicjava.lang.StringgetContents()
```

---

setContents
```
public void setContents(java.lang.String contents)
```

---

getCur_temp
```
public float getCur_temp()
```

---

setCur_temp
```
public void setCur_temp(float cur_temp)
```

---

getDestinationId
```
publicintgetDestinationId()
```

---

setDestinationId
```
public void setDestinationId(int destinationId)
```

---

getFinal_temp
```
public float getFinal_temp()
```

---

setFinal_temp
```
public void setFinal_temp(float final_temp)
```

---

getFromDest
```
public java.awt.geom.Point2D.Float getFromDest()
```

---

setFromDest
```
public void setFromDest(java.awt.geom.Point2D.Float fromDest)
```

---

getId
```
publicintgetId()
```

---

setId
```
public void setId(int id)
```

---

isPerforming
```
publicbooleanisPerforming()
```

---

setPerforming
```
public void setPerforming(boolean performing)
```

---

getQuantity
```
public float getQuantity()
```

---

setQuantity
```
public void setQuantity(float quantity)
```

---

getSourceId
```
publicintgetSourceId()
```

---

setSourceId
```
public void setSourceId(int sourceId)
```

---

getToDest
```
public java.awt.geom.Point2D.Float getToDest()
```

---

setToDest
```
public void setToDest(java.awt.geom.Point2D.Float toDest)
```

---

setType
```
public void setType(java.lang.String type)
```

---

getType
```
publicjava.lang.StringgetType()
```

---

execute
```
public void execute()
```

---

exportXml
```
public void exportXml()
```

---

saveState
```
public void saveState()
```

# Package ekshiksha.vclab.lab

ekshiksha.vclab.lab

## 2.1 Class VirtualChemLab

```
java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ java.awt.Panel
└ java.applet.Applet
└ javax.swing.JApplet
└ ekshiksha.vclab.lab.VirtualChemLab
```

All Implemented Interfaces:
java.awt.event.ActionListener, java.awt.event.ComponentListener, java.awt.event.MouseListener, java.awt.event.MouseMotionListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible, javax.swing.RootPaneContainer

## Constructor Detail

VirtualChemLab
publicVirtualChemLab()

## Method Detail

init
public void init()
Initialization method that will be called after the applet is loaded. into the browser. In this method, all resources such as images will be loaded. and all support objects instantiated. After this method, the applet is ready to run.
Store mode:
to read the chemical list from the xml file using XMLReader
add a new bottle for each chemical(a maximum of 10 at a time will be shown)
add apparatus of several types apparatus list:
beaker
pipette
burette
test tube
flask
burner
Overrides:
`init` in class `java.applet.Applet`

---

initVariablesForPositioning
public void initVariablesForPositioning()
This method initializes various variables for proper positioning of equipments on workbench.

---

createAndShowGUI
public void createAndShowGUI()
This method creates the user-interface depending on the current mode. Adds the JComponents to the applet depending on the current mode.
Sets the initial sizes of the JComponents
Adds JMenubar options like save, open,switchmode,etc.
Add the buttons for play,pause,show/hide blackboard,etc.
adds the stopwatch(perform mode)
initializes various variables for proper positioning of equipments on cupboard(store mode).
Adds the blackboard(perform/play mode)

---

setMode
public void setMode(int mode)
This method sets the mode for blackboard
Parameters:
`mode` - : The mode to which the blackboard is to be set

---

getMode
publicintgetMode()
This method is called to get value of data element mode.

---

processMode
public void processMode()
This method is called whenever there is a change in mode.

start
public void start()
Overrides:
start in class java.applet.Applet

componentResized
public void componentResized(java.awt.event.ComponentEvent e)
Resizes the various positioning variables, and panels whenever the applet is resized
Specified by:
componentResized in interface java.awt.event.ComponentListener
Parameters:
e - Event generated whenever the applet is resized

componentMoved
public void componentMoved(java.awt.event.ComponentEvent e)
Specified by:
componentMoved in interface java.awt.event.ComponentListener

componentShown
public void componentShown(java.awt.event.ComponentEvent e)
Specified by:
componentShown in interface java.awt.event.ComponentListener

componentHidden
public void componentHidden(java.awt.event.ComponentEvent e)
Specified by:
componentHidden in interface java.awt.event.ComponentListener

actionPerformed
public void actionPerformed(java.awt.event.ActionEvent e)
Abstract method for Action Listener
Specified by:
actionPerformed in interface java.awt.event.ActionListener
Parameters:
e - Event corresponding to button or Jmenu option
STORE MODE:
open: to open a pre-existing experiment set-up file
save: to save the experiment set-up file
chemBottleFromList: to add To add a new chemical which is not present in the list. The user is prompted to enter the name, formula and colour of the chemical(as RGB)
chemBottleToList:Adding a chemical from the chemical list shown as a drop-down list.
perform: to switch to perform mode
PERFORM MODE:
board: to show/hide blackboard
play:to switch to play mode
open: to open an experiment set-up file
save: to save the activities performed
store: to switch to set-up mode (only for 'teacher' user)
PLAY MODE
open: to open an experiment set-up file

store: to switch to set-up mode (only for 'teacher' user)
perform: to switch to perform mode

---

removeAllObjects
public void removeAllObjects()

---

promptForHeader
public void promptForHeader()
This method prompts the teacher to enter header information. Header fields : String title; String author_Name; String level; int marks String description; String student_Name; String instructions

---

mouseEntered
public void mouseEntered(java.awt.event.MouseEvent me)
Specified by:
`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited
public void mouseExited(java.awt.event.MouseEvent me)
Specified by:
`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed
public void mousePressed(java.awt.event.MouseEvent me)
Specified by:
`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased
public void mouseReleased(java.awt.event.MouseEvent me)
This method is implementation of the abstract method mouseReleased
Specified by:
`mouseReleased` in interface `java.awt.event.MouseListener`
Parameters:
`me` -
On releasing the chemical bottle/equipment in the workbench, it is automatically placed in a suitable position on shelf or table.
In case of chemical bottle, the user is asked the concentration of the chemical bottle he wants to provide
In case of equipments, the user is asked to choose one of the various sizes available for that particular equipment.

---

mouseClicked
public void mouseClicked(java.awt.event.MouseEvent me)
Specified by:
`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseMoved
public void mouseMoved(java.awt.event.MouseEvent me)
This method is implementation of the abstract method mouseMoved for MouseListener
Specified by:
`mouseMoved` in interface `java.awt.event.MouseMotionListener`
Parameters:

me -
Prompts for header of the experiment if not yet filled(Store/set-up mode)
Change the border colour of the equipment in cupboard when mouse is brought on it

---

mouseDragged
public void mouseDragged(java.awt.event.MouseEvent me)
This method is implementation of the abstract method mouseDragged
Specified by:
mouseDragged in interface java.awt.event.MouseMotionListener
Parameters:
me -
Moves equipments when mouse is dragged
create a new equipment of the same type if the current equipment reaches at the border of the cupboard and workbench

ekshiksha.vclab.lab

# 2.2 Class WorkBench

java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent
└ javax.swing.JPanel
└ ekshiksha.vclab.lab.WorkBench
All Implemented Interfaces:
java.awt.event.ActionListener, java.awt.event.ComponentListener, java.awt.event.MouseListener, java.awt.event.MouseMotionListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.lang.Runnable, java.util.EventListener, javax.accessibility.Accessible

| Constructor Detail |
|---|

WorkBench
publicWorkBench(StopWatchRunner watch)

| Method Detail |
|---|

getb_rx
public float getb_rx()

---

getb_ry
public float getb_ry()

---

getMode
publicintgetMode()

---

setMode
public void setMode(int mode)

---

paintComponent
public void paintComponent(java.awt.Graphics g)
This method is used to paint the components and equipments of the virtual lab
Overrides:
paintComponent in class javax.swing.JComponent

Parameters:
g - Graphics object This method over rides the paintComponent method of the JPanel class. It automatically gets called once on execution. And it also gets called when repaint method is called. This method is used draw the different parts of workbench by calling the drawLab(Graphics g) method. It also initializes the initial origins of all equipments in the equipment list and calls their drawEquipment(Graphics g) method.Hence this method controls the drawing of graphics of the project.

---

drawLab
public void drawLab(java.awt.Graphics g)
This method is called by paintComponent to draw the different parts of the lab.
Parameters:
g - Graphics object To draw the different parts of the lab it calls different methods for the different parts i.e. table(using drawTable(Graphics g) of Table class),shelf(using drawShelf(Graphics g) of Shelf class), basin(using drawBasin(Graphics g) of Basin class),Tiles(using fillTiles(Graphics g) of Tiles class).

---

getTableCoordinates
publicjava.awt.PointgetTableCoordinates(int idx)

---

getRandomPoint
publicjava.awt.PointgetRandomPoint()

---

mouseEntered
public void mouseEntered(java.awt.event.MouseEvent me)
Specified by:
mouseEntered in interface java.awt.event.MouseListener

---

mouseClicked
public void mouseClicked(java.awt.event.MouseEvent me)
Specified by:
mouseClicked in interface java.awt.event.MouseListener

---

mouseExited
public void mouseExited(java.awt.event.MouseEvent me)
Specified by:
mouseExited in interface java.awt.event.MouseListener

---

mousePressed
public void mousePressed(java.awt.event.MouseEvent me)
This method extracts coordinates of where the mouse has been pressed
Specified by:
mousePressed in interface java.awt.event.MouseListener
Parameters:
me - MouseEvent object This mouse Listener works only in the Store and Perform experiment mode and not in the Demonstration mode. After extracting coordinates of where the mouse has been pressed, it checks if that point is within the Equipment List using the function public booleancontains(Point2D.Float c,Point2D.Float org,floatwt,floatht). If it finds that the selected Equipment object is rotated by an angle due to previous Activity like pouring, it re-rotates it back to zero degrees. It also sets the value of currentIndex of the selected Equipment from the arraylistEquipments.

---

mouseReleased
public void mouseReleased(java.awt.event.MouseEvent me)
This mouseListener detects a click and release of the mouse button
Specified by:
`mouseReleased` in interface `java.awt.event.MouseListener`
Parameters:
`me` - MouseEvent object This listener is the most important one as it has the function of identifying if an Activity is about to take place. By iterating across the list Equipments, it checks using public booleancontains(Point2D.Float c,Point2D.Float org,floatwt,floatht) whether 2 equipment regions are coinciding. a)If the equipment dragged is a Pipette it sets it to middle top region of the other equipment. A JOptionPane appears and takes the user input of Yes/No and the act is added to Activities list. b) When a beaker/Flask is dragged to the lower region of a Burette it sets the former equipment to lower middle region of the Burette and Enables the Burette pour buttons to be functional. c) Otherwise after performing some constraint checks it sets the currentIndex equipment to top left corner of the coinciding equipment. d) If Equipment regions are not coinciding but the equipment is dragged to an empty space the Activity is registered as a Move. e) If Equipment is dragged to Basin Region and the user inputs yes to a JOptionPane prompt the Activity is registered as a Wash.

---

mouseMoved
public void mouseMoved(java.awt.event.MouseEvent me)
This Mouse Listener detects pure motion of the mouse without button clicks * @param me MouseEvent object The main purpose of this mouse Listener is to change the border color parameter of an equipment to red when the mouse is hovered within the equipment. It also sets the color to green if the equipment is hovered in the wash basin region of the Lab. It also provides the user control to rotate equipment objects and filling or de-filing the Pipette when pouring such that each time the mouse is moved an increment in angle or change in volume of the Pipette takes place.
Specified by:
`mouseMoved` in interface `java.awt.event.MouseMotionListener`

---

mouseDragged
public void mouseDragged(java.awt.event.MouseEvent me)
This mouseListener is used to detect dragging motion of the mouse * @param me MouseEvent object The major action of this mouse Listener is to keep redrawing each equipment every instant as it is dragged across the Lab by changing the originShift coordinates of them to the getX() getY() coordinates of the mouse. The pour activity from Burette to Beaker or Flask is added to the Activities arrayList within this block.
Specified by:
`mouseDragged` in interface `java.awt.event.MouseMotionListener`

---

componentResized
public void componentResized(java.awt.event.ComponentEvent e)
This method is used to resize the components of the lab on resizing the window
Specified by:
`componentResized` in interface `java.awt.event.ComponentListener`
Parameters:
`ComponentEvent` - object This method is a method of the ComponentListener interface and gets called whenever the Workbench is re sized. It sets the new origins of the equipments on the Workbench according to the new size of the Workbench.The other methods of ComponentListener interface are not being used.

---

componentMoved
public void componentMoved(java.awt.event.ComponentEvent e)
Specified by:
`componentMoved` in interface `java.awt.event.ComponentListener`

---

componentShown
public void componentShown(java.awt.event.ComponentEvent e)
Specified by:
`componentShown` in interface `java.awt.event.ComponentListener`

---

componentHidden
public void componentHidden(java.awt.event.ComponentEvent e)
Specified by:
`componentHidden` in interface `java.awt.event.ComponentListener`

---

actionPerformed
public void actionPerformed(java.awt.event.ActionEvent e)
This method is for capturing actions for the Pour Prompt
Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`
Parameters:
`e` - ActionEvent object This block of code is to perform actions when the buttons of Yes or No on the JFrame pour prompt is pressed by the user. The frame is set visible in mouseReleased when a pour is about to take place. The user inputs the desired quantity and the particular equipment is rotated accordingly and consistent volume changes are made across two equipments. Filling from a burette and other Pour activities are registered within here.Also checks to prevent a user from entering garbage values, negative quantities, quantities which are not available in the equipment or quantities which will cause one equipment to overflow are implemented with unique JOptionPane warning messages being displayed for each.

---

run
public void run()
Specified by:
`run` in interface `java.lang.Runnable`

ekshiksha.vclab.lab

# 2.3 Class Play_module

java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent
└ javax.swing.JPanel
└ ekshiksha.vclab.lab.Play_module
All Implemented Interfaces:
java.awt.event.ActionListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.lang.Runnable, java.util.EventListener, javax.accessibility.Accessible

| Method Detail |
|---|

actionPerformed
public void actionPerformed(java.awt.event.ActionEvent ae)

this function changes the various boolean values to determine the flow of the experiment namely play, pause, skip step, backward step.
Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

move
publicint move(Activity activity,
int a)
This function moves the given equipment from the initial point to the final point, it uses the repaint technique to create an animation effect. the function can be paused, forwarded or back-stepped.
Parameters:
`activity` - : the object of activity currently being performed
`a` - : id of the equipment upon which activity is being performed.
Returns:
: the completion mode of the function.

---

pour
publicint pour(Activity activity,
int a)
This function moves the given equipment from the initial point to the destination equipment, then it pours the quantity specified into the destination object It uses the repaint technique to create an animation effect. All the different combinations of equipments have been considered the function can be paused, forwarded or back-stepped.
Parameters:
`activity` - : the object of activity currently being performed.
`a` - : id of the equipment upon which activity is being performed.
Returns:
: the completion mode of the function.

---

wash
publicint wash(Activity activity,
int a)
This function moves the given equipment from the initial point to the basin, the equipment is then emptied(i.e washed) it uses the repaint technique to create an animation effect. the function can be paused, forwarded or back-stepped.
Parameters:
`activity` - : the object of activity currently being performed
`a` - : id of the equipment upon which activity is being performed.
Returns:
: the completion mode of the function.

---

perform_exp
public void perform_exp()
This function begins the infinite loop to constantly call the required activity. even after the experiment is over the experiment can be back-stepped and viewed again.

---

run
public void run()
Begins the execution of the experiment.
Specified by:
`run` in interface `java.lang.Runnable`

ekshiksha.vclab.lab

# 2.4 Class Cupboard

java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent
└ javax.swing.JPanel
└ ekshiksha.vclab.lab.Cupboard
All Implemented Interfaces:
java.awt.event.ComponentListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible

## Method Detail

setDrawThree
public void setDrawThree(boolean q)

---

paintComponent
public void paintComponent(java.awt.Graphics g)
This method calls the initEquipment() and drawEquipment() functions for each equipment in the cupboardEquipmentList.
Overrides:
`paintComponent` in class `javax.swing.JComponent`

---

drawLab
public void drawLab(java.awt.Graphics g)
This method is used to draw tiles and tables.

---

addApparatus
public void addApparatus(Equipment bk)

---

setShowTable
public void setShowTable(boolean st)

---

getListSize
publicintgetListSize()

---

componentResized
public void componentResized(java.awt.event.ComponentEvent ce)
This method is used resize all the equipments whenever the Applet window is resized.
Specified by:
`componentResized` in interface `java.awt.event.ComponentListener`

---

componentMoved
public void componentMoved(java.awt.event.ComponentEvent ce)
Specified by:
`componentMoved` in interface `java.awt.event.ComponentListener`

---

componentShown
public void componentShown(java.awt.event.ComponentEvent ce)

Specified by:
componentShown in interface java.awt.event.ComponentListener

---

componentHidden
public void componentHidden(java.awt.event.ComponentEvent ce)
Specified by:
componentHidden in interface java.awt.event.ComponentListener

# Package ekshiksha.vclab.parser

ekshiksha.vclab.parser

## 3.1 Class XMLReader

java.lang.Object
└ ekshiksha.vclab.parser.XMLReader

| Method Detail |
|---|

getHead
publicHeadergetHead()
To return the object of the header class.
Returns:
head: object of header class.

---

getEvaluation
publicEvaluationgetEvaluation()

---

getObservation
publicObservationgetObservation()

---

getEquipmentList
publicjava.util.ArrayList<Equipment>getEquipmentList()
Function to return the array list for store.
Returns:
storeequip : array list for store.

---

removeAll
public void removeAll()

ekshiksha.vclab.parser

## 3.2 Class WriteXMLFile

java.lang.Object
└ ekshiksha.vclab.parser.WriteXMLFile

| Constructor Detail |
|---|

WriteXMLFile
publicWriteXMLFile(Header header,
java.util.ArrayList<Equipment> setup,
int type,
java.io.File file)

Constructor for Store to write XML.
Parameters:
`header` - : object of Header class.
`setup` - : Array list of equipment class.
`type` - : Array list of activity class.
`file` - : XML file to be written.

---

WriteXMLFile
publicWriteXMLFile(Header header,
java.util.ArrayList<Equipment> setup,
java.util.ArrayList<Activity> perform,
Observation observation,
Evaluation evaluation,
java.io.File file)
Constructor for Store to write XML.
Parameters:
`header` - : object of Header class.
`setup` - : Array list of equipment class.
`perform` - : Array list of activity class.
`observation` - : Object of Observation class.
`evaluation` - : Object of Evaluation class.
`file` - : XML file to be written.

# Package ekshiksha.vclab.equipment

ekshiksha.vclab.equipment

## 4.1 Class Beaker

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.Beaker
All Implemented Interfaces:
EquipmentType

| Constructor Detail |
|---|

Beaker
public Beaker(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,
float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
The constructor for WorkBench.
Parameters:
`id` - : id of the equipment.

eqname - : type of the equipment.
origin - : initial position of the equipment.
quantity - : initial quantity.
size - : size of the equipment.
contents - : name of the chemical present initially in the equipment.
contentColor - : colour of the chemical present initially in the equipment.
strength - : strength of the chemical present initially in the equipment.
environ - : object of WorkBench.java

---

Beaker
public Beaker(java.awt.geom.Point2D.Float origin,
Cupboard e)
The constructor for store.
Parameters:
origin - : initial position of the equipment.
e - : object of Cupboard.java

Method Detail

initEquipment
public void initEquipment()
this function redraws the equipment for the present size of the window and the current angle.
Overrides:
initEquipment in class Equipment

---

rotate
public void rotate(double theta)
This function rotates the equipment by the given angle(positive or negative)(in degrees).
Overrides:
rotate in class Equipment
Parameters:
theta - : the angle by which the equipment has to be rotated.

---

fill
public void fill(float quant)
This function fills the equipment by the given quantity.
Overrides:
fill in class Equipment
Parameters:
quant - : quantity to be filled.

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
This function draws lines between the points plotted in chooseSize(or initEquipment), thereby drawing the equipment.
Overrides:
drawEquipment in class Equipment
Parameters:
g - : Object of Graphics class.

---

setQuantity
public void setQuantity(float quant)

Sets the quantity to the given value.
Overrides:
setQuantity in class Equipment
Parameters:
`quant` - : the value to which the quantity of the equipment has to be set.

---

getIterationSteps
public float getIterationSteps(float quant,
Equipment eq)
this function calculates the angle to which the equipment has to be rotated for pouring the given quantity.
Overrides:
getIterationSteps in class Equipment
Parameters:
`quant` - : quantity to be poured.
`eq` - : object of the destination equipment.
Returns:
: angle, to which the equipment has to be rotated for pouring the given quantity.

---

drawContents
public void drawContents(java.awt.Graphics g)
This function draws the liquid contained in the equipment according to the quantity and the angle. it is also responsible for reducing the quantity of the current equipment and filling the destination equipment by the appropriate quantity.
Parameters:
`g` - : Object of Graphics class.

---

setSize
public void setSize(int size)
Sets the size of the equipment.
Overrides:
setSize in class Equipment
Parameters:
`size` - : size of the equipment.

ekshiksha.vclab.equipment

# 4.2 Class Bottle

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.Bottle
All Implemented Interfaces:
EquipmentType

| Constructor Detail |
|---|

Bottle
public Bottle(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,

float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
This is the WorkBench constructor for the Bottle object.
Parameters:
`id` - equipment id
`eqname` - equipment name
`origin` - equipment origin
`quantity` - equipment quantity
`size` - equipment size
`contents` - contents of equipment
`contentColor` - color of content
`strength` - strength of equipment
`environ` - environ
Throws:
`IOException`

---

Bottle
public Bottle(java.awt.geom.Point2D.Float origin,
Cupboard e)
This is the Cupboard constructor for the Bottle object.
Parameters:
`origin` - equipment origin
`e` - object of store
Throws:
`IOException`

## Method Detail

initEquipment
public void initEquipment()
This Method initializes the bottle This function sets the coordinates of the equipment object in the array of points coords according to the angle parameter of the object.
Overrides:
initEquipment in class Equipment

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
This method draws equipment
Overrides:
drawEquipment in class Equipment
Parameters:
`g` - Graphics Object This function draws straight lines across the points stored in coords to generate the equipment image and also sets the string content label beside the Equipment.

---

rotate
public void rotate(double theta)
This method is used to rotate the equipment
Overrides:

rotate in class Equipment
Parameters:
`theta` - angle This geometrically rotates any given equipment object across the desired angle which has to be submitted in terms of degrees.

---

drawContents
public void drawContents(java.awt.Graphics g)
This method draws the contents of the equipment
Parameters:
`g` - Graphics object This function uses a polygon object to fill the equipment according to its current fraction and capacity by initializing the corner points of the polygon chemical suitably. It has blocks for various fractions of liquid to be filled for the equipment. It finally uses a fill polygon method.

---

fill
public void fill(float quant)
This method is used to fill the equipment
Overrides:
fill in class Equipment
Parameters:
`quant` - Quantity This function updates the fraction parameter and hence quantity of the equipment when the user calls it. Volumes are usually denoted in ml.

---

getIterationSteps
public float getIterationSteps(float quant,
Equipment eq)
this function calculates the angle to which the equipment has to be rotated for pouring the given quantity.
Overrides:
getIterationSteps in class Equipment
Parameters:
`quant` - : quantity to be poured.
`eq` - : object of the destination equipment.
Returns:
: angle, to which the equipment has to be rotated for pouring the given quantity.

---

setQuantity
public void setQuantity(float quant)
Overrides:
setQuantity in class Equipment

---

setSize
public void setSize(int size)
Overrides:
setSize in class Equipment

ekshiksha.vclab.equipment

# 4.3 Class Burette

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.Burette

All Implemented Interfaces:
EquipmentType, java.awt.event.ActionListener, java.util.EventListener

## Constructor Detail

Burette
public Burette(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,
float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
The constructor for WorkBench.
Parameters:
`id` - : id of the equipment.
`eqname` - : type of the equipment.
`origin` - : initial position of the equipment.
`quantity` - : initial quantity.
`size` - : size of the equipment.
`contents` - : name of the chemical present initially in the equipment.
`contentColor` - : colour of the chemical present initially in the equipment.
`strength` - : strength of the chemical present initially in the equipment.
`environ` - : object of WorkBench.java

---

Burette
public Burette(java.awt.geom.Point2D.Float origin,
Cupboard e)
The constructor for store.
Parameters:
`origin` - : initial position of the equipment.
`e` - : object of Cupboard.java

## Method Detail

initEquipment
public void initEquipment()
this function redraws the equipment for the present size of the window and the current angle.
Overrides:
initEquipment in class Equipment

---

Round
public static float Round(float Rval,
int Rpl)
function for rounding a number.
Parameters:
`Rval` : - value to be rounded off.
`Rpl` : - number of places.
Returns:
: rounded value.

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
This function draws lines between the points plotted in chooseSize(or initEquipment), thereby drawing the equipment.
Overrides:
drawEquipment in class Equipment
Parameters:
g - : Object of Graphics class.

---

setQuantity
public void setQuantity(float quant)
Sets the quantity to the given value.
Overrides:
setQuantity in class Equipment
Parameters:
quant - : the value to which the quantity of the equipment has to be set.

---

fill
public void fill(float quant)
This function fills the equipment by the given quantity.
Overrides:
fill in class Equipment
Parameters:
quant - : quantity to be filled.

---

drawContents
public void drawContents(java.awt.Graphics g)
This function draws the liquid contained in the equipment according to the quantity. it is also responsible for reducing the quantity of the current equipment and filling the destination equipment by the appropriate quantity.
Parameters:
g - : Object of Graphics class.

---

pour
public void pour(java.awt.Graphics g)
removes 1ml of liquid from the current equipment and fills it in the destination equipment.
Overrides:
pour in class Equipment
Parameters:
g - : Object of Graphics class.

---

getIterationSteps
public float getIterationSteps(float quant,
Equipment eq)
this function calculates the number of times pour has to be called for pouring the given quantity.
Overrides:
getIterationSteps in class Equipment
Parameters:
quant - : quantity to be poured.
eq - : object of the destination equipment.
Returns:

: no. of steps.

---

actionPerformed
public void actionPerformed(java.awt.event.ActionEvent ae)
Used in perform mode, to pour into another equipment on button click.
Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`
Parameters:
`ae` - : Object of action event.

---

setSize
public void setSize(int size)
Sets the size of the equipment.
Overrides:
setSize in class Equipment
Parameters:
`size` - : size of the equipment

ekshiksha.vclab.equipment

# 4.4 Class Burner

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.Burner
All Implemented Interfaces:
EquipmentType, java.io.Serializable, java.lang.Runnable

| Constructor Detail |
|---|

Burner
public Burner(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,
float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
throwsjava.io.IOException
Parameters:
`id` - equipment id
`eqname` - equipment name
`origin` - equipment origin
`quantity` - equipment quantity
`size` - equipment size
`contents` - contents of equipment
`contentColor` - color of content
`strength` - strength of equipment
`environ` - environ

Throws:
`java.io.IOException`

---

Burner
public Burner(java.awt.geom.Point2D.Float origin,
Cupboard e)
throwsjava.io.IOException
Parameters:
`origin` - equipment origin
`e` - object of store
Throws:
`java.io.IOException`

## Method Detail

getPreferredSize
publicjava.awt.DimensiongetPreferredSize()
for getting preferred size
Returns:
dimension

---

setIntensity
public void setIntensity(double d)
for setting intensity
Parameters:
`d` - intensity

---

getIntensity
public double getIntensity()
for getting intensity
Returns:
intensity

---

setState
public void setState(boolean b)
for setting state
Parameters:
`b` - booleanonState

---

getState
publicbooleangetState()
for getting state
Returns:
onState

---

initEquipment
public void initEquipment()
This Method initializes the pipette This function sets the coordinates of the equipment object in the array of points coords according to the angle parameter of the object.
Overrides:
initEquipment in class Equipment

---

getApparatusHeight
public float getApparatusHeight()
for getting apparatus height
Returns:
height

---

getApparatusWidth
public float getApparatusWidth()
for getting apparatus width
Returns:
width

---

getOrigin
public java.awt.geom.Point2D.Float getOrigin()
for getting origin
Returns:
originShift

---

flame
public void flame(int time)
this function is responsible for automatic flames
Parameters:
`time` - this shows the time in seconds for which we want the burner to burn

---

getMouthLeft
public java.awt.geom.Point2D.Float getMouthLeft()
for getting mouth left
Returns:
mouthLeft

---

getMouthRight
public java.awt.geom.Point2D.Float getMouthRight()
for getting mouth right
Returns:
mouthRight

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
this function is responsible for drawing equipment
Overrides:
drawEquipment in class Equipment
Parameters:
`g` - object of graphics class

---

setLineColor
public void setLineColor(java.awt.Color col)
for setting line color
Parameters:
`col` - color

---

getFlameHeight

publicintgetFlameHeight()
for getting flame height
Returns:
flameHeight

---

run
public void run()
Specified by:
`run` in interface `java.lang.Runnable`

---

draw
public void draw(java.awt.Graphics g)

ekshiksha.vclab.equipment

# 4.5 Class Flask

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.Flask
All Implemented Interfaces:
EquipmentType

| Constructor Detail |
|---|

Flask
public Flask(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,
float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
The constructor for WorkBench.
Parameters:
`id` - : id of the equipment.
`eqname` - : type of the equipment.
`origin` - : initial position of the equipment.
`quantity` - : initial quantity.
`size` - : size of the equipment.
`contents` - : name of the chemical present initially in the equipment.
`contentColor` - : colour of the chemical present initially in the equipment.
`strength` - : strength of the chemical present initially in the equipment.
`environ` - : object of WorkBench.java

---

Flask
public Flask(java.awt.geom.Point2D.Float origin,
Cupboard e)
The constructor for store.
Parameters:

origin - : initial position of the equipment.
e - : object of Cupboard.java

Method Detail

initEquipment
public void initEquipment()
this function redraws the equipment for the present size of the window and the current angle.
Overrides:
initEquipment in class Equipment

---

rotate
public void rotate(double theta)
This function rotates the equipment by the given angle(positive or negative)(in degrees).
Overrides:
rotate in class Equipment
Parameters:
theta - : the angle by which the equipment has to be rotated.

---

fill
public void fill(float quant)
This function fills the equipment by the given quantity.
Overrides:
fill in class Equipment
Parameters:
quant - : quantity to be filled.

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
This function draws lines between the points plotted in chooseSize(or initEquipment), thereby drawing the equipment.
Overrides:
drawEquipment in class Equipment
Parameters:
g - : Object of Graphics class.

---

setQuantity
public void setQuantity(float quant)
Sets the quantity to the given value.
Overrides:
setQuantity in class Equipment
Parameters:
quant - : the value to which the quantity of the equipment has to be set.

---

getIterationSteps
public float getIterationSteps(float quant,
Equipment eq)
this function calculates the angle to which the equipment has to be rotated for pouring the given quantity.
Overrides:
getIterationSteps in class Equipment
Parameters:

quant - : quantity to be poured.
eq - : object of the destination equipment.
Returns:
: angle, to which the equipment has to be rotated for pouring the given quantity.

---

drawContents
public void drawContents(java.awt.Graphics g)
This function draws the liquid contained in the equipment according to the quantity and the angle. it is also responsible for reducing the quantity of the current equipment and filling the destination equipment by the appropriate quantity.
Parameters:
g - : Object of Graphics class.

---

setSize
public void setSize(int size)
Sets the size of the equipment.
Overrides:
setSize in class Equipment
Parameters:
size - : size of the equipment.

ekshiksha.vclab.equipment

# 4.6 Class Pipette

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.Pipette
All Implemented Interfaces:
EquipmentType

---

public class Pipette
extendsEquipment
this class is responsible for equipment pipette. this class can do the following:
initialization of pipette.
drawing of pipette.

| Constructor Detail |
|---|

Pipette
public Pipette(java.awt.geom.Point2D.Float origin,
WorkBench e)

---

Pipette
public Pipette(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,
float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
This is the WorkBench constructor for the Bottle object.

Parameters:
`id` - equipment id
`eqname` - equipment name
`origin` - equipment origin
`quantity` - equipment quantity
`size` - equipment size
`contents` - contents of equipment
`contentColor` - color of content
`strength` - strength of equipment
`environ` - environ
Throws:
`IOException`

---

Pipette
public Pipette(java.awt.geom.Point2D.Float origin,
Cupboard e)
This is the Cupboard constructor for the Bottle object.
Parameters:
`origin` - equipment origin
`e` - object of store
Throws:
`IOException`

**Method Detail**

initEquipment
public void initEquipment()
This Method initializes the pipette This function sets the coordinates of the equipment object in the array of points coords according to the angle parameter of the object.
Overrides:
initEquipment in class Equipment

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
This method draws the equipment
Overrides:
drawEquipment in class Equipment
Parameters:
`g` - Graphics object This function draws straight lines across the points stored in coords to generate the equipment image. If the pipette is full then the contents are drawn.For filling the equipment this method calls the fill method of this equipment and calls pour method of the equipment that is pouring into this equipment.For pouring the equipment this method calls the fill method of this equipment and calls fill method of the equipment that is pouring into this equipment.

---

fill
public void fill(java.awt.Graphics g,
Equipment eq)
This method fills the pipette
Overrides:
fill in class Equipment
Parameters:
`g` - Graphics object

eq - Equipment from which pipette is being filled This method fills the equipment increasing quantity step wise. It uses the Path2D.Float object to fill the equipment.Accordingly the quantity and fraction of the equipment are changed.

---

pour
public void pour(java.awt.Graphics g,
Equipment eq)
This pipette is used to pour liquid from pipette
Overrides:
pour in class Equipment
Parameters:
g - Graphics object
eq - Equipment to which liquid is poured This method fills the equipment with decreasing quantity step wise. It uses the Path2D.Float object to fill the equipment.Accordingly the quantity and fraction of the equipment are changed.

---

setOriginShift
public void setOriginShift(java.awt.geom.Point2D.Float pt)
this method sets the originShift
Overrides:
setOriginShift in class Equipment
Parameters:
pt - new origin

---

getIterationSteps
public float getIterationSteps(float quantity,
Equipment eq)
This method returns the no of steps used for filling/pouring for a particular quantity
Overrides:
getIterationSteps in class Equipment
Parameters:
quantity - quantity
eq - Equipment
Returns:
This function is used by the Demonstration module for stepwise filling and pouring into and from an equipment.

---

setQuantity
public void setQuantity(float quantity)
Overrides:
setQuantity in class Equipment

---

setSize
public void setSize(int size)
This method sets size
Overrides:
setSize in class Equipment
Parameters:
size - size required

ekshiksha.vclab.equipment

# 4.7 Class TestTube

java.lang.Object
└ ekshiksha.vclab.equipment.Equipment
└ ekshiksha.vclab.equipment.TestTube
All Implemented Interfaces:
EquipmentType

## Constructor Detail

TestTube
publicTestTube(int id,
java.lang.String eqname,
java.awt.geom.Point2D.Float origin,
float quantity,
int size,
java.lang.String contents,
java.awt.Color contentColor,
float strength,
WorkBench environ)
The constructor for WorkBench.
Parameters:
`id` - : id of the equipment.
`eqname` - : type of the equipment.
`origin` - : initial position of the equipment.
`quantity` - : initial quantity.
`size` - : size of the equipment.
`contents` - : name of the chemical present initially in the equipment.
`contentColor` - : colour of the chemical present initially in the equipment.
`strength` - : strength of the chemical present initially in the equipment.
`environ` - : object of WorkBench.java

---

TestTube
publicTestTube(java.awt.geom.Point2D.Float origin,
Cupboard e)
The constructor for store.
Parameters:
`origin` - : initial position of the equipment.
`e` - : object of Cupboard.java

## Method Detail

initEquipment
public void initEquipment()
this function redraws the equipment for the present size of the window and the current angle.
Overrides:
initEquipment in class Equipment

---

rotate
public void rotate(double theta)
This function rotates the equipment by the given angle(positive or negative)(in degrees).
Overrides:

rotate in class Equipment
Parameters:
theta - : the angle by which the equipment has to be rotated.

---

fill
public void fill(float quant)
This function fills the equipment by the given quantity.
Overrides:
fill in class Equipment
Parameters:
quant - : quantity to be filled.

---

drawEquipment
public void drawEquipment(java.awt.Graphics g)
This function draws lines between the points plotted in chooseSize(or initEquipment), thereby drawing the equipment.
Overrides:
drawEquipment in class Equipment
Parameters:
g - : Object of Graphics class.

---

setQuantity
public void setQuantity(float quant)
Sets the quantity to the given value.
Overrides:
setQuantity in class Equipment
Parameters:
quant - : the value to which the quantity of the equipment has to be set.

---

getIterationSteps
public float getIterationSteps(float quant,
Equipment eq)
this function calculates the angle to which the equipment has to be rotated for pouring the given quantity.
Overrides:
getIterationSteps in class Equipment
Parameters:
quant - : quantity to be poured.
eq - : object of the destination equipment.
Returns:
: angle, to which the equipment has to be rotated for pouring the given quantity.

---

drawContents
public void drawContents(java.awt.Graphics g)
This function draws the liquid contained in the equipment according to the quantity and the angle. it is also responsible for reducing the quantity of the current equipment and filling the destination equipment by the appropriate quantity.
Parameters:
g - : Object of Graphics class.

---

setSize

public void setSize(int size)
Sets the size of the equipment.
Overrides:
setSize in class Equipment
Parameters:
`size` - : size of the equipment.

---

# Challenges Faced

- Interpretation of the previously submitted code by last year's interns as it lacked sufficient documentation
- Identifying Interaction of equipments to start pour activities,especially for burette and pipette
- Showing stepwise filling of equipments in play module
- Moving objects across cupboard from store to workbench by creating duplicate objects
- Implementing Polygon fill algorithms for filling quantity of each equipment

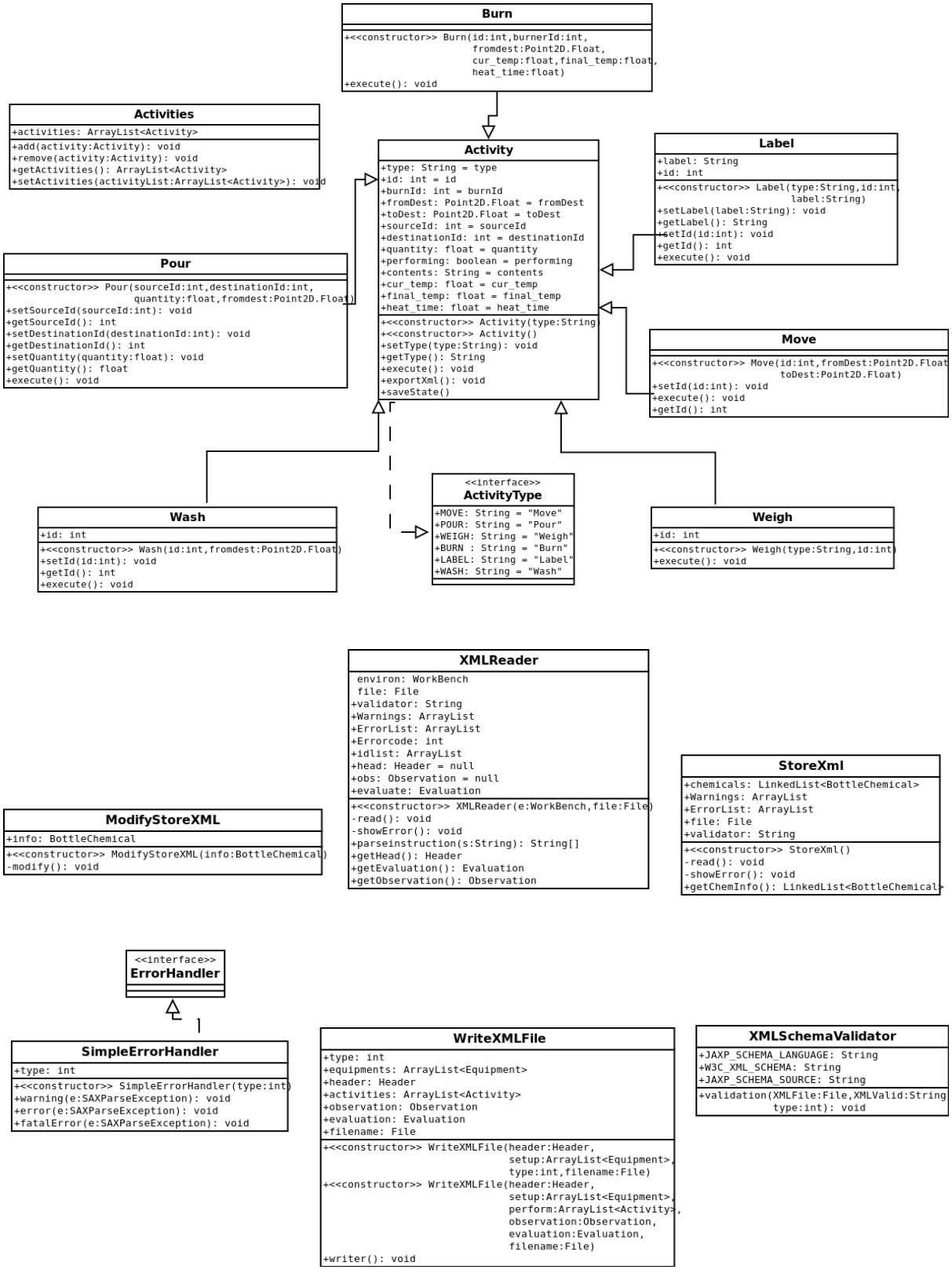
# Limitations

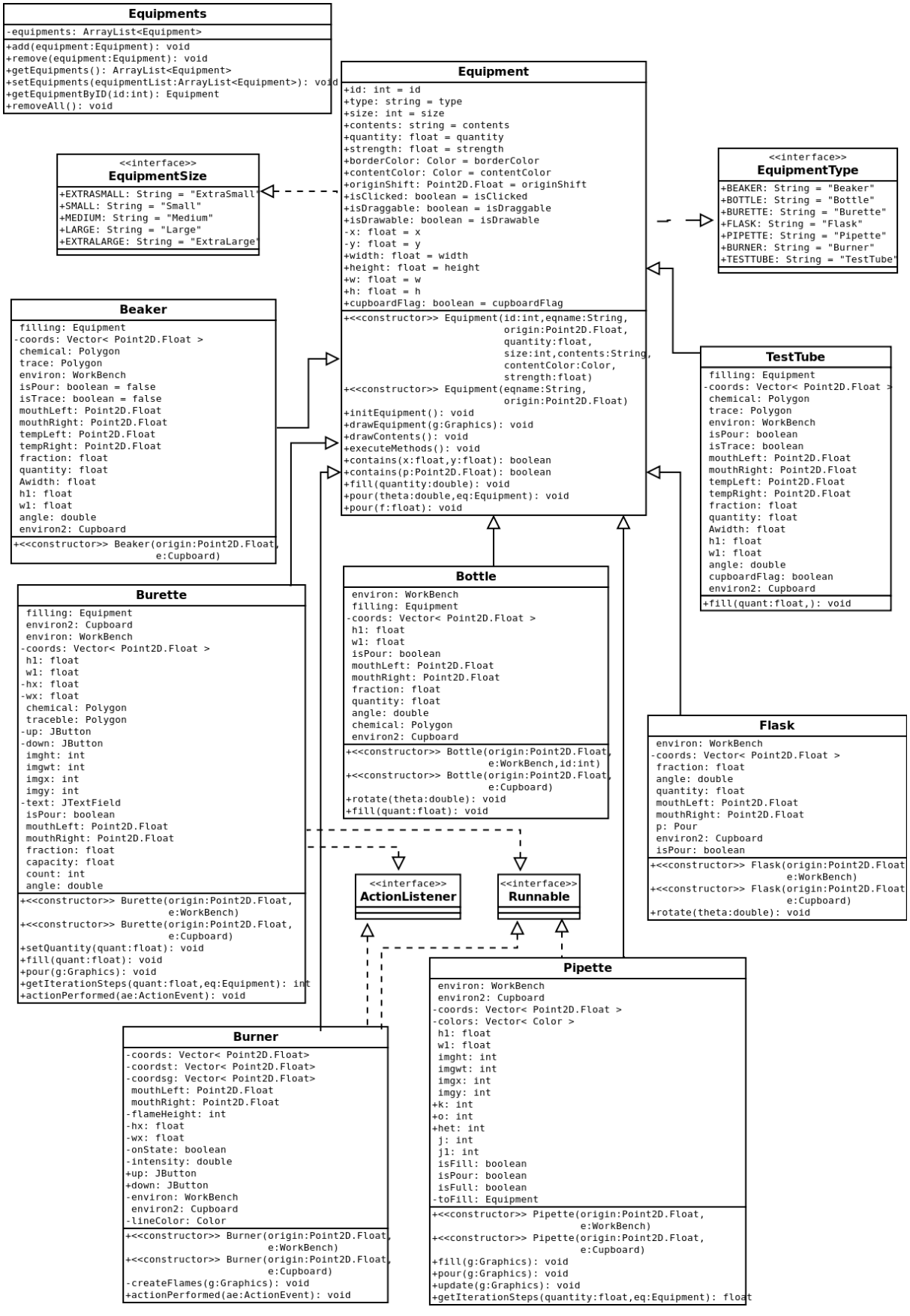
- Does not contain any knowledge for chemical contents of the Lab but relies on sample experiments for display
- There are a few constraints as to how equipment must be dragged or moved around to perform some activities
- Backstep functionality of play module is stepwise and does not give relevance to time taken to perform the activity but to order in which they are performed

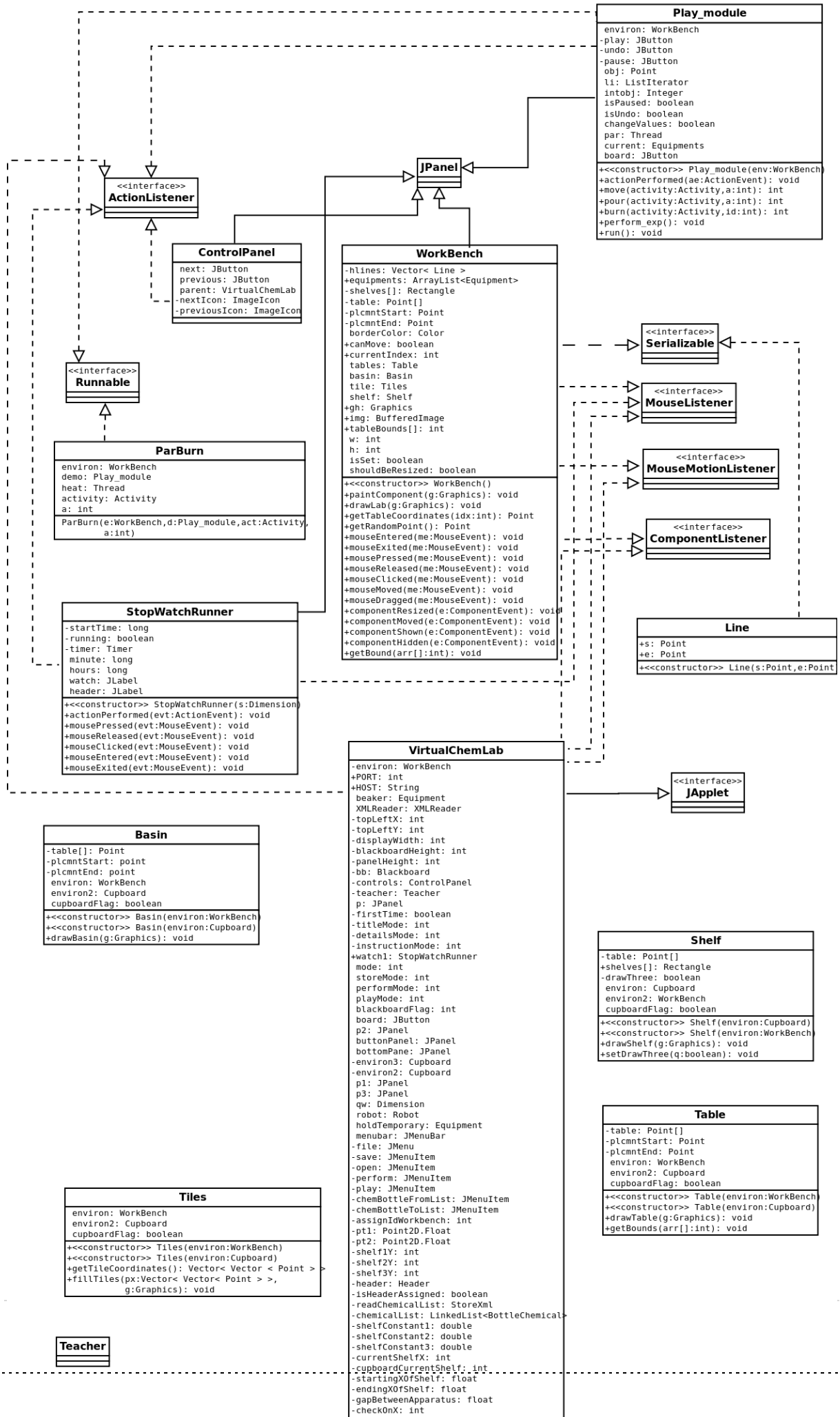
# Future Scope

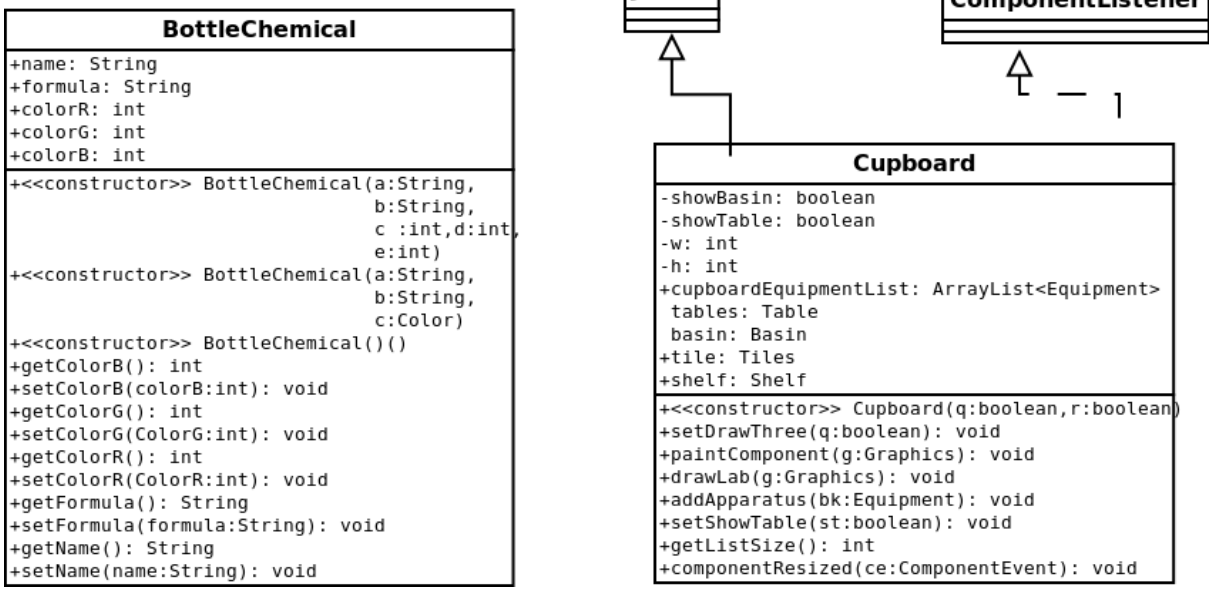
- All chemical knowledge and physical properties like color or temperature may be integrated in a separate class
- Volumetric and titration calculations can be performed
- At a more advanced stage graphics for chemical reaction changes such as effervescence or boiling may be added

# APPENDIX : CLASS DIAGRAMS OF VIRTUAL CHEMISTRY LAB




Equipments
-equipments: ArrayList<Equipment>
+add(equipment:Equipment): void
+remove(equipment:Equipment): void
+getEquipments(): ArrayList<Equipment>
+setEquipments(equipmentList:ArrayList<Equipment>): void
+getEquipmentByID(id:int): Equipment
+removeAll(): void
Equipment
+id: int = id
+type: string = type
+size: int = size
+contents: string = contents
+quantity: float = quantity
+strength: float = strength
+borderColor: Color = borderColor
+contentColor: Color = contentColor
+originShift: Point2D.Float = originShift
+isClicked: boolean = isClicked
+isDraggable: boolean = isDraggable
+isDrawable: boolean = isDrawable
-x: float = x
-y: float = y
+width: float = width
+height: float = height
+w: float = w
+h: float = h
+cupboardFlag: boolean = cupboardFlag
+<<constructor>> Equipment(id:int,eqname:String, origin:Point2D.Float, quantity:float, size:int,contents:String, contentColor:Color, strength:float)
+<<constructor>> Equipment(eqname:String, origin:Point2D.Float)
+initEquipment(): void
+drawEquipment(g:Graphics): void
+drawContents(): void
+executeMethods(): void
+contains(x:float,y:float): boolean
+contains(p:Point2D.Float): boolean
+fill(quantity:double): void
+pour(theta:double,eq:Equipment): void
+pour(f:float): void
<<interface>>
EquipmentSize
+EXTRASMALL: String = "ExtraSmall"
+SMALL: String = "Small"
+MEDIUM: String = "Medium"
+LARGE: String = "Large"
+EXTRALARGE: String = "ExtraLarge"
<<interface>>
EquipmentType
+BEAKER: String = "Beaker"
+BOTTLE: String = "Bottle"
+BURETTE: String = "Burette"
+FLASK: String = "Flask"
+PIPETTE: String = "Pipette"
+BURNER: String = "Burner"
+TESTTUBE: String = "TestTube"
Beaker
filling: Equipment
-coords: Vector< Point2D.Float >
chemical: Polygon
trace: Polygon
environ: WorkBench
isPour: boolean = false
isTrace: boolean = false
mouthLeft: Point2D.Float
mouthRight: Point2D.Float
tempLeft: Point2D.Float
tempRight: Point2D.Float
fraction: float
quantity: float
Awidth: float
h1: float
w1: float
angle: double
environ2: Cupboard
+<<constructor>> Beaker(origin:Point2D.Float, e:Cupboard)
TestTube
filling: Equipment
-coords: Vector< Point2D.Float >
chemical: Polygon
trace: Polygon
environ: WorkBench
isPour: boolean
isTrace: boolean
mouthLeft: Point2D.Float
mouthRight: Point2D.Float
tempLeft: Point2D.Float
tempRight: Point2D.Float
fraction: float
quantity: float
Awidth: float
h1: float
w1: float
angle: double
cupboardFlag: boolean
environ2: Cupboard
+fill(quant:float,): void
Bottle
environ: WorkBench
filling: Equipment
-coords: Vector< Point2D.Float >
h1: float
w1: float
isPour: boolean
mouthLeft: Point2D.Float
mouthRight: Point2D.Float
fraction: float
quantity: float
angle: double
chemical: Polygon
environ2: Cupboard
+<<constructor>> Bottle(origin:Point2D.Float, e:WorkBench,id:int)
+<<constructor>> Bottle(origin:Point2D.Float, e:Cupboard)
+rotate(theta:double): void
+fill(quant:float): void
Burette
filling: Equipment
environ2: Cupboard
environ: WorkBench
-coords: Vector< Point2D.Float >
h1: float
w1: float
-hx: float
-wx: float
chemical: Polygon
traceble: Polygon
-up: JButton
-down: JButton
imght: int
imgwt: int
imgx: int
imgy: int
-text: JTextField
isPour: boolean
mouthLeft: Point2D.Float
mouthRight: Point2D.Float
fraction: float
capacity: float
count: int
angle: double
+<<constructor>> Burette(origin:Point2D.Float, e:WorkBench)
+<<constructor>> Burette(origin:Point2D.Float, e:Cupboard)
+setQuantity(quant:float): void
+fill(quant:float): void
+pour(g:Graphics): void
+getIterationSteps(quant:float,eq:Equipment): int
+actionPerformed(ae:ActionEvent): void
Flask
environ: WorkBench
-coords: Vector< Point2D.Float >
fraction: float
angle: double
quantity: float
mouthLeft: Point2D.Float
mouthRight: Point2D.Float
p: Pour
environ2: Cupboard
isPour: boolean
+<<constructor>> Flask(origin:Point2D.Float e:WorkBench)
+<<constructor>> Flask(origin:Point2D.Float e:Cupboard)
+rotate(theta:double): void
<<interface>>
ActionListener
<<interface>>
Runnable
Pipette
environ: WorkBench
environ2: Cupboard
-coords: Vector< Point2D.Float >
-colors: Vector< Color >
h1: float
w1: float
imght: int
imgwt: int
imgx: int
imgy: int
+k: int
+o: int
+het: int
j: int
j1: int
isFill: boolean
isPour: boolean
isFull: boolean
-toFill: Equipment
+<<constructor>> Pipette(origin:Point2D.Float, e:WorkBench)
+<<constructor>> Pipette(origin:Point2D.Float, e:Cupboard)
+fill(g:Graphics): void
+pour(g:Graphics): void
+update(g:Graphics): void
+getIterationSteps(quantity:float,eq:Equipment): float
Burner
-coords: Vector< Point2D.Float>
-coordst: Vector< Point2D.Float>
-coordsg: Vector< Point2D.Float>
mouthLeft: Point2D.Float
mouthRight: Point2D.Float
-flameHeight: int
-hx: float
-wx: float
-onState: boolean
-intensity: double
+up: JButton
+down: JButton
-environ: WorkBench
environ2: Cupboard
-lineColor: Color
+<<constructor>> Burner(origin:Point2D.Float, e:WorkBench)
+<<constructor>> Burner(origin:Point2D.Float, e:Cupboard)
-createFlames(g:Graphics): void
+actionPerformed(ae:ActionEvent): void


Play_module
environ: WorkBench
-play: JButton
-undo: JButton
-pause: JButton
obj: Point
li: ListIterator
intobj: Integer
isPaused: boolean
isUndo: boolean
changeValues: boolean
par: Thread
current: Equipments
board: JButton
+<<constructor>> Play_module(env:WorkBench)
+actionPerformed(ae:ActionEvent): void
+move(activity:Activity,a:int): int
+pour(activity:Activity,a:int): int
+burn(activity:Activity,id:int): int
+perform_exp(): void
+run(): void
<<interface>>
ActionListener
JPanel
ControlPanel
next: JButton
previous: JButton
parent: VirtualChemLab
-nextIcon: ImageIcon
-previousIcon: ImageIcon
WorkBench
-hlines: Vector< Line >
+equipments: ArrayList<Equipment>
-shelves[]: Rectangle
-table: Point[]
-plcmntStart: Point
-plcmntEnd: Point
borderColor: Color
+canMove: boolean
+currentIndex: int
tables: Table
basin: Basin
tile: Tiles
shelf: Shelf
+gh: Graphics
+img: BufferedImage
+tableBounds[]: int
w: int
h: int
isSet: boolean
shouldBeResized: boolean
+<<constructor>> WorkBench()
+paintComponent(g:Graphics): void
+drawLab(g:Graphics): void
+getTableCoordinates(idx:int): Point
+getRandomPoint(): Point
+mouseEntered(me:MouseEvent): void
+mouseExited(me:MouseEvent): void
+mousePressed(me:MouseEvent): void
+mouseReleased(me:MouseEvent): void
+mouseClicked(me:MouseEvent): void
+mouseMoved(me:MouseEvent): void
+mouseDragged(me:MouseEvent): void
+componentResized(e:ComponentEvent): void
+componentMoved(e:ComponentEvent): void
+componentShown(e:ComponentEvent): void
+componentHidden(e:ComponentEvent): void
+getBound(arr[]:int): void
<<interface>>
Serializable
<<interface>>
MouseListener
<<interface>>
MouseMotionListener
<<interface>>
ComponentListener
<<interface>>
Runnable
ParBurn
environ: WorkBench
demo: Play_module
heat: Thread
activity: Activity
a: int
ParBurn(e:WorkBench,d:Play_module,act:Activity,
a:int)
StopWatchRunner
-startTime: long
-running: boolean
-timer: Timer
minute: long
hours: long
watch: JLabel
header: JLabel
+<<constructor>> StopWatchRunner(s:Dimension)
+actionPerformed(evt:ActionEvent): void
+mousePressed(evt:MouseEvent): void
+mouseReleased(evt:MouseEvent): void
+mouseClicked(evt:MouseEvent): void
+mouseEntered(evt:MouseEvent): void
+mouseExited(evt:MouseEvent): void
Line
+s: Point
+e: Point
+<<constructor>> Line(s:Point,e:Point
VirtualChemLab
-environ: WorkBench
+PORT: int
+HOST: String
beaker: Equipment
XMLReader: XMLReader
-topLeftX: int
-topLeftY: int
-displayWidth: int
-blackboardHeight: int
-panelHeight: int
-bb: Blackboard
-controls: ControlPanel
-teacher: Teacher
p: JPanel
-firstTime: boolean
-titleMode: int
-detailsMode: int
-instructionMode: int
+watch1: StopWatchRunner
mode: int
storeMode: int
performMode: int
playMode: int
blackboardFlag: int
board: JButton
p2: JPanel
buttonPanel: JPanel
bottomPane: JPanel
-environ3: Cupboard
-environ2: Cupboard
p1: JPanel
p3: JPanel
qw: Dimension
robot: Robot
holdTemporary: Equipment
menubar: JMenuBar
-file: JMenu
-save: JMenuItem
-open: JMenuItem
-perform: JMenuItem
-play: JMenuItem
-chemBottleFromList: JMenuItem
-chemBottleToList: JMenuItem
-assignIdWorkbench: int
-pt1: Point2D.Float
-pt2: Point2D.Float
-shelf1Y: int
-shelf2Y: int
-shelf3Y: int
-header: Header
-isHeaderAssigned: boolean
-readChemicalList: StoreXml
-chemicalList: LinkedList<BottleChemical>
-shelfConstant1: double
-shelfConstant2: double
-shelfConstant3: double
-currentShelfX: int
-cupboardCurrentShelf: int
-startingXOfShelf: float
-endingXOfShelf: float
-gapBetweenApparatus: float
-checkOnX: int
<<interface>>
JApplet
Basin
-table[]: Point
-plcmntStart: point
-plcmntEnd: point
environ: WorkBench
environ2: Cupboard
cupboardFlag: boolean
+<<constructor>> Basin(environ:WorkBench)
+<<constructor>> Basin(environ:Cupboard)
+drawBasin(g:Graphics): void
Shelf
-table: Point[]
+shelves[]: Rectangle
-drawThree: boolean
environ: Cupboard
environ2: WorkBench
cupboardFlag: boolean
+<<constructor>> Shelf(environ:Cupboard)
+<<constructor>> Shelf(environ:WorkBench)
+drawShelf(g:Graphics): void
+setDrawThree(q:boolean): void
Table
-table: Point[]
-plcmntStart: Point
-plcmntEnd: Point
environ: WorkBench
environ2: Cupboard
cupboardFlag: boolean
+<<constructor>> Table(environ:WorkBench)
+<<constructor>> Table(environ:Cupboard)
+drawTable(g:Graphics): void
+getBounds(arr[]:int): void
Tiles
environ: WorkBench
environ2: Cupboard
cupboardFlag: boolean
+<<constructor>> Tiles(environ:WorkBench)
+<<constructor>> Tiles(environ:Cupboard)
+getTileCoordinates(): Vector< Vector < Point > >
+fillTiles(px:Vector< Vector< Point > >,
g:Graphics): void
Teacher


BottleChemical
+name: String
+formula: String
+colorR: int
+colorG: int
+colorB: int
+<<constructor>> BottleChemical(a:String,
b:String,
c :int,d:int,
e:int)
+<<constructor>> BottleChemical(a:String,
b:String,
c:Color)
+<<constructor>> BottleChemical()()
+getColorB(): int
+setColorB(colorB:int): void
+getColorG(): int
+setColorG(ColorG:int): void
+getColorR(): int
+setColorR(ColorR:int): void
+getFormula(): String
+setFormula(formula:String): void
+getName(): String
+setName(name:String): void
JPanel
<<interface>>
ComponentListener
1
Cupboard
-showBasin: boolean
-showTable: boolean
-w: int
-h: int
+cupboardEquipmentList: ArrayList<Equipment>
tables: Table
basin: Basin
+tile: Tiles
+shelf: Shelf
+<<constructor>> Cupboard(q:boolean,r:boolean)
+setDrawThree(q:boolean): void
+paintComponent(g:Graphics): void
+drawLab(g:Graphics): void
+addApparatus(bk:Equipment): void
+setShowTable(st:boolean): void
+getListSize(): int
+componentResized(ce:ComponentEvent): void

**Debug**

```
+ON: boolean = false
```

**ErrorCode**

```
+SUCCESS:  int = 0
+WARNINGS_FOUND: int = -1
+ERROR_FOUND: int = -2
```

**Constants**

```
+DEFAULT_OUTLINE_COLOR_FOR_LAB: Color = Color.BLACK
+TILE_SEPEEATION_AT_LOWEST_LEVEL: double = (3.0/28.0
+TILT_ANGLE: int = 50
+TILT_ANGLE_WALL: int = 50
+FLOOR_LIMIT : double = 5
+TILE_COLOR_FIRST: Color = new Color( 50, 50, 50 )
+TILE_COLOR_SECOND: Color = new Color(230, 230, 230)
+LENGTH_EDGE_HORIZONTAL_LINE: double = (3.0/70.0)
+TABLE_HEIGHT_FROM_EDGE_STARTINGS: double = (1.0/7.0
+FRACTION_FIRST: double = 0.15
+FRACTION_SECOND : double = 0.7
+TABLE_CROSS_SECION_WIDTH: double = (3.0/70.0)
+EDGE_LEG_DISTANCE: double = (15.0/512.0)
+LEG_WIDTH: double = (5.0/128.0)
+FRONT_LEG_HEIGTH: double = (1.0/7.0)
+LEG_COLOR: Color = new Color(140, 80, 60 )
+LEG_EFFECT: Color = new Color( 151, 105, 79 )
+DISTANCE_RIGHT_EDGE_BASIN: double = (5.0/256.0)
+DISTANCE_BOTTOM_EDGE_BASIN: double = (5.0/128.0)
+BACK_LENGTH_BASIN: double = (15.0/256.0)
+BASIN_WIDTH_X: double = (5.0/128.0)
+BASIN_WIDTH_Y: double = (2.0/35.0)
+TAP_THICKNESS_X: double = (1.0/128.0)
+TAP_THICKNESS_Y: double = (2.0/175.0)
+TAP_HEIGHT: double = 0.1
+TAP_WIDTH: double = (5.0/128.0)
+TAP_COLOR: Color = new Color( 192, 192, 192 )
+WALL_COLOR: Color = new Color(245, 229, 176 )
+SHELF_LENGTH: int = 295
+TOP_SHELF_MARGIN: double = (3.0/35.0)
+BOTTOM_SHELF_MARGIN: int = 0
+SHELF_DX: double = (25.0/512.0)
+SHELF_DY: double = (3.0/70.0)
+TOP_BOTTOM_COLOR: Color = new Color( 170, 170, 170
+MID_SHELF_SLIDER: double = 1.8
+APPARATUS_SEPERATION: int = 6
+TEXT_APP_TOP_DISTANCE: int = 10
+FONT_SIZE: int = 9
+FONT_NAME: String = "Monotype Corsiva"
+SEPERATION_BASINLEFT_PLCMNTEND: int = 20
+DEMO: int = 1
+ EXP: int = 2
+FLAME_COLOR: Color = Color.ORANGE
+FLAME_INTENSITY_DECREAMENT: double =  0.1
+MAX_FLAME_INTENSITY: int = 3
+MAX_FLAME_HEIGHT: int = 20
+DEFAULT_DELAY: int = 30
+POUR_ACTIVITY: int = 1
+HEAT_ACTIVITY: int = 2
```

REFERENCES


1. Oracle JavaDocs: http://docs.oracle.com/javase/6/docs/api/

2. www.stackoverflow.com: For coding related queries.

3. Java How to Program, 9/e : Paul Deitel, Harvey Deitel

4. G. Booch, J. Rumbaugh, and I. Jacobson, I. The Unified Modeling Language User Guide. Addison-Wesley, 1999 .

5. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995 .

6. F. Buschmann et al. Pattern Oriented Software Architecture, Volume 1: A System of Patterns. John Wiley and Sons, 1996.

7. M. Shaw and D. Garlan. Software Architecture: Perspectives on an Emerging Discipline. Prentice-Hall, 1996