Project: Mindmaps for JabRef

Possible mentors: Oliver Kopp, Carl Christian Snethlage, Dominic Voigt

Organization: JabRef e.V.

Google Summer of Code `21

About Me

Name and Contact information:

- **Name: TAIJASI KAVERI**
- **Time Zone: IST (UTC+5:30 hours)**
- **Github id:** Taijasi-Kaveri
- **Email:** kaveri.taijasi8@gmail.com
- **LinkedIn:** Taijasi Kaveri
- **Medium (for Blogging):** Taijasi Kaveri
- **Mobile: (INDIA)** +91-9572937030
- **Place of residing:** Noida, India

Education:

- **University:** Dr. A.P.J. Abdul Kalam Technical University
- **College:** J.S.S. Academy of Technical Education, Noida
- **Degree:** Bachelor of Technology (B.Tech)
- **Field of Study:** Computer Science Engineering
- **Current Year:** 3rd
- **CGPA:** 8.24 (out of 10)
- **Expected Graduation Year:** July 2022

About the Project

Abstract:

JabRef organizes library entries in list-way. It is an open source software to manage references and the full text of papers. The native file format used by JabRef is BibTeX, the standard LaTeX bibliography format. The project "Mindmaps for JabRef" will help in organising the library entries in a mindmap way such that each item is annotated with more mindmap nodes. Thus, making it easy for the user to organize the references.

Deliverables:

The idea is to organize library entries in a mindmap way by:

- Creating basic setup of the application with Java
- Using JavaFX to craft mindmaps
- Using **Apache NetBeans 11.3** environment for building a reactive application on the JVM
- Using either
    - Socket Programming
    - RMI
    - JavaGroups

  to connect to JabRef to retrieve and store data.

Implementation Plan:

Task 1: To create basic setup of the application in Java

This is the basic setup required for the application

- a main class to use the graph library you create
- a graph with a data model
- easy adding and removing of nodes and edges.
- a zoomable scrollpane
- a layout algorithm for the graph.

The application instantiates the graph, adds cells and connects them via edges. Also, this application organizes library entries in a mindmap way and each node is annotated with more mindmap nodes.

Task 2: Crafting Mindmaps using JavaFX

For the library entries to be displayed in a mindmap way, we will be using JavaFX.

This application is based on Graph Theory .

Here are few examples on how our codebase would be like:

Cell.java

This is a basic code to create cells in the application.

```
package com.fxgraph.graph;

import java.util.ArrayList;
import java.util.List;

import javafx.scene.Node;
import javafx.scene.layout.Pane;

public class Cell extends Pane {
```

```
String cellId;

List<Cell> children = new ArrayList<>();
List<Cell> parents = new ArrayList<>();

Node view;

public Cell(String cellId) {
    this.cellId = cellId;
}

public void addCellChild(Cell cell) {
    children.add(cell);
}

public List<Cell> getCellChildren() {
    return children;
}

public void addCellParent(Cell cell) {
    parents.add(cell);
}

public List<Cell> getCellParents() {
    return parents;
}

public void removeCellChild(Cell cell) {
    children.remove(cell);
}

public void setView(Node view) {

    this.view = view;
    getChildren().add(view);

}

public Node getView() {
    return this.view;
    }

    public String getCellId() {
        return cellId;
    }
}
```

The cells can be further instantiated into

i. Rectangle
ii. Triangle

ZoomableScrollPane.java

Code Source: [pixel duke]

```
public class ZoomableScrollPane extends ScrollPane{
  Group zoomGroup;
  Scale scaleTransform;
  Node content;
  (…)
  public ZoomableScrollPane(Node content)
  {
    this.content = content;
    Group contentGroup = new Group();
    zoomGroup = new Group();
    contentGroup.getChildren().add(zoomGroup);
    zoomGroup.getChildren().add(content);
    setContent(contentGroup);
    scaleTransform = new Scale(scaleValue, scaleValue, 0, 0);
    zoomGroup.getTransforms().add(scaleTransform);
    (…)
  }
  (...)
}
```

Edge.java

The edges can be Curve but in this example I have preferred line.

```
package com.fxgraph.graph;

import javafx.scene.Group;
import javafx.scene.shape.Line;

public class Edge extends Group {

    protected Cell source;
    protected Cell target;

    Line line;

    public Edge(Cell source, Cell target) {

        this.source = source;
        this.target = target;

        source.addCellChild(target);
        target.addCellParent(source);

        line = new Line();
        line.startXProperty().bind(
source.layoutXProperty().add(source.getBoundsInParent().getWidth()         /
2.0));
        line.startYProperty().bind(
source.layoutYProperty().add(source.getBoundsInParent().getHeight()        /
2.0));

        line.endXProperty().bind(                        target.layoutXProperty().add(
target.getBoundsInParent().getWidth() / 2.0));
        line.endYProperty().bind(                        target.layoutYProperty().add(
target.getBoundsInParent().getHeight() / 2.0));

        getChildren().add( line);

    }

    public Cell getSource() {
        return source;
    }

    public Cell getTarget() {
        return target;
    }

}
```

IDE to be used:

i. Apache NetBeans 11.3

Creating a Maven project using NetBeans IDE is comparatively easy. As the only work required id installing the Java correctly. All the configuration we need is in a single POM XML file. NetBeans natively understands Maven, so all we would need to do is open the file through NetBeans' open project menu action, and start coding.

Task 3: Connecting the application to JabRef for retrieval and storage of data

For connecting our application to JabRef , we can use either of the three methods;

1. *Socket Programming*
2. *RMI*
3. *JavaGroups*

***1.Socket** :*

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

2.RMI:

The good old RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

Server creates a local registry:

```
ServerImpl extends UnicastRemoteObject implements Server

rmiRegistry = LocateRegistry.createRegistry(2020);
rmiRegistry.bind("server", aServerImpl);
```

Client looks it up with an RMI url

```
String url = "rmi://localhost:2020/server";
Server server = (Server) Naming.lookup(url);
```

3.JavaGroups :

JavaGroups is a group communication toolkit written entirely in Java. It is based on IP multicast, but extends it with

1. reliability and
2. group membership.

The API of JavaGroups is very simple (similar to a UDP socket) and is always the same, regardless of how the underlying protocol stack is composed. To be able to send/receive messages, a *channel* has to be created. The reliability of a channel is specified as a string, which then causes the creation of the underlying protocol stack. The example below creates a channel and sends/receives 1 message

```
String props="UDP:PING:FD:STABLE:NAKACK:UNICAST:" +
             "FRAG:FLUSH:GMS:VIEW_ENFORCER:STATE_TRANSFER:QUEUE";
Message send_msg, recv_msg;
Channel channel=new JChannel(props);

channel.Connect("MyGroup");
send_msg=new Message(null, null, "Hello world");
channel.Send(send_msg);
recv_msg=(Message)channel.Receive(0);
System.out.println("Received " + recv_msg);

channel.Disconnect();
channel.Close();
```

Timeline

Milestones:

Before the official coding time

★ **Before May 17**

- To familiarize myself completely with JabRef's software functionality and architecture.
- To familiarize myself with JavaFX.
- To do self coding with Java to improve my further understanding and ease of use it.
- Make demo apps to get hands-on with the environment and languages that I will be using in the main project.

★ **May 17 – June 7 (Community Bonding Period)**

- Familiarize with the codebase and the community, the version control system and the documentation.
- Discuss with the mentor, a plan, to implement more functionality into the application.
- This period can also be used to further refine, discuss and plan the components of the proposal which are not very clear.
- During this period I will remain in constant touch with my mentor and the JabRef community. I will remain active on Gitter channel and Mailing lists to discuss and finalize on the modifications.

- Thus with the help of my mentor I will become absolutely clear about my future goals,the final implementations that need to be done as well as the approach that I will follow.

Official coding period starts (June 7)

★ **June 7 – June 17**
- To familiarize myself with the IDE.
- Discuss and finalise the basic setup of the Java application.
- Implement the code to create the application.

★ **June 18 – June 30**
- Writing code in JavaFX to craft mindmaps.

★ **(July 1 – July 15) – Tentative dates for university exams**

★ **Mid-term evaluations (July 12- July 16)**

★ **July 16 – July 22**
- Incorporate the change suggested by mentors post the mid evaluations.
- Testing the overall functioning of application.

★ **July 23 – July 31**
- Add support for the application to work with JabRef.
- Connect the two applications and check for their working.

★ **Aug 1 – Aug 15**
- Making further changes in the code to improve the Functionality, Exception handling, Bug Removal.

- To be in constant touch with the JabRef's developers and to let them know about our progress.
- For Documentation of code

★ **Final Evaluations and Submit code (Aug 16 – Aug 23)**

Apart from the above-mentioned schedule, I will be:

Blogging:

In the important phases of GSoC, I will be writing blog posts about my progress. In this blog I will be sharing my ideas, writing about advantages and disadvantages of different implementations, and asking for feedback (though, I will also be doing that on the mailing list).

Communication:

During the coding period, I will remain in constant communication with my mentors. Also, I will push daily commits and send weekly pull requests. I will also be sharing progress being made weekly and make sure that the outcome is as it was planned to be.

GSoC

1. **Have you used JabRef before? Do you know about BibTeX/biblatex? (Formulate in your own words)**

   Answer. No, I haven't used JabRef before.

   As of now**,** I do not have enough knowledge about BibTeX/biblatex but I am doing vigorous research from my side so that I am familiar enough with their functioning.

2. **Why do you want to contribute to JabRef in GSOC?**

   Answer. Before that, I have never heard of anything like Citation and Bibliography Reference Manager Tool. The concept of JabRef software interests me a lot. And also, I got to know about different reference formats like Native BibTeX and BibLaTeX. Morover, the projects are exciting as well. For eg., Mindmaps, the project I am working on helps in creating automatic mindmaps for different library entries.

   For me this is a great opportunity to collaborate with JabRef community, to learn a lot of new things and to create something useful.

3. **Did you participate in GSOC or any other project before? If yes, for which project(s).**

   Answer. No, I didn't participate in GSoC or any other open source project before.

4. **Are you participating in any other organisations other than JabRef ?**

   Answer. No, JabRef is the sole organization I am working with this year.

Why I am a good fit?

- ✓ Dedicated undergraduate well known for sincerety and responsibility towards my work.
- ✓ Able to quickly grasp the concepts of any programming language.
- ✓ I do my best to complete the assigned work before its deadline.
- ✓ I always strive to take up challenges which helps me learning new things. This is the reason I chose this project which is based on Maven dependency, which was a foreign subject for me at that time.
- ✓ Morover, I have intermediate level of experience with Java.
  (Here's the certificate Sololearn Certificate for Java)
- ✓ Great interest in learning JavaFX
  (I am learning JavaFX and its fundamentals from various online sites and Youtube tutorials.
  This will help me in developing the project efficiently and on time.

Commitments

In the month of July**,** I will be having my endsems exams (the date of which has not been confirmed by my university yet) thus I won't be able to continue coding in that particular period (which will last for 10-15 days). To compensate my unavailability during exams, I will be contributing in the month of June for 4-5 hours per day. If I miss any task, I will work extra hours to finish it before deadline.

Other than that, I will keep contributing for 2-3 hours per day.

Eligibility

I am eligible to participate in the Google Summer of Code. For any queries, clarifications or further explanations, feel free to contact (kaveri.taijasi8@gmail.com).

****