```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionListener;
 4
 5 public class MenuBar {
 6     private final ActionListener MenuBarCalled;
 7
 8
 9     MenuBar(ActionListener MBCall) {
10         MenuBarCalled = MBCall;
11     }
12
13
14     protected JButton makeNavigationButton(String txt
   , String actionCommand, String toolTipText) {
15         Font fnt = new Font("Times New Roman", Font.
   PLAIN, 18);
16
17         //Create and initialise the MenuButton.
18         JButton MenuButton = new JButton();
19         MenuButton.setText(txt);
20         MenuButton.setFont(fnt);
21         MenuButton.setToolTipText(toolTipText);
22         MenuButton.setActionCommand(actionCommand);
23         MenuButton.addActionListener(MenuBarCalled);
24
25         return MenuButton;
26     }
27 }
28
```

```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5
 6 public class Task_gui extends JFrame implements
   ActionListener {
 7     JFrame TKframe = new JFrame("Task");
 8     JPanel tkpnl = new JPanel();
 9     JPanel btnpnl = new JPanel();
10
11     Font tkfnt = new Font("Times New Roman", Font.
   PLAIN, 18);
12
13     JButton btnTKSave = new JButton("Save Task");
14
15
16     JLabel lblDescript = new JLabel("Task Description
   ");
17     JLabel lblDuration = new JLabel("Task Duration");
18     JLabel lblleads = new JLabel("Task leads from");
19
20     JTextField txtDescript = new JTextField();
21     JTextField txtDuration = new JTextField();
22     JTextField txtleads = new JTextField();
23     Coursework main;
24
25     // public static void main(String[] args) {
26     //     Task tk = new Task();
27     // }
28
29     public Task_gui(Coursework m){
30         Task();
31         main = m;
32     }
33
34     private void Task(){
35         lblDescript.setFont(tkfnt);
36         lblDuration.setFont(tkfnt);
37         lblleads.setFont(tkfnt);
38
39         btnTKSave.setFont(tkfnt);
40         btnTKSave.addActionListener(this);
41         btnTKSave.setActionCommand("ok");
```

```
42          btnpnl.add(btnTKSave);
43
44          tkpnl.setLayout(new GridLayout(3,2, 15,15));
45          tkpnl.add(lblDescript);
46          tkpnl.add(txtDescript);
47          tkpnl.add(lblDuration);
48          tkpnl.add(txtDuration);
49          tkpnl.add(lblleads);
50          tkpnl.add(txtleads);
51
52          TKframe.add(btnpnl, BorderLayout.CENTER);
53          TKframe.add(tkpnl, BorderLayout.NORTH);
54
55          TKframe.setDefaultCloseOperation(JFrame.
    DISPOSE_ON_CLOSE);
56          TKframe.setExtendedState(JFrame.
    MAXIMIZED_BOTH);
57          TKframe.pack();
58          TKframe.setVisible(true);
59
60      }
61
62
63      @Override
64      public void actionPerformed(ActionEvent e) {
65          if ("ok".equals(e.getActionCommand())){
66              main.data.getCurrentProject().addTask(
    txtDescript.getText(),Integer.parseInt(txtDuration.
    getText()), txtleads.getText());
67              Coursework c = new Coursework();
68              main.frame.setVisible(false);
69          }
70          TKframe.setVisible(false);
71      }
72  }
73
```

```
 1 import javax.swing.*;
 2 import javax.swing.event.ListSelectionEvent;
 3 import javax.swing.event.ListSelectionListener;
 4 import java.awt.*;
 5 import java.awt.event.ActionEvent;
 6 import java.awt.event.ActionListener;
 7 import java.io.IOException;
 8 import java.time.LocalDate;
 9 import java.util.ArrayList;
10
11 import classes.*;
12
13 //Note to self
14 //Calendar, gridlayout look, Gantt chart placement
   and actionperformed needs to be do.
15 //cannot retrieve the calendar, mess around with
   gridbagconstraint to make Project, Team and Task GUI
   frame look more pleasing.
16 //Gantt chart need to be added once it is created
17 //actionperformed only works for toolbar variables
   not any other.
18 //Fix this Abidon once you wake up
19 public class Coursework extends JFrame implements
   ActionListener{
20
21     public static programHandler data =
   programHandler.getInstance();
22
23     critical_path path = new critical_path(data);
24
25     MenuBar mb = new MenuBar(this);
26
27     JFrame frame = new JFrame();
28     JPanel westpnl = new JPanel(new GridBagLayout());
29     JPanel centerpnl = new JPanel(new GridBagLayout
   ());
30     JPanel eastpnl = new JPanel(new GridBagLayout());
31
32     JToolBar toolBar = new JToolBar();
33     JButton btnNewProject = new JButton("New Project"
   );
34     JButton btnAddTeam = new JButton("Add Team");
35     JButton btnAssignPerson = new JButton("Assign
   Team to Task");
```

```
36      JButton btnSave = new JButton("Save");
37      JButton btnCreateTeam = new JButton("Create Team"
    );
38      JButton btncritPath = new JButton("Critical Path"
    );
39
40      Font LabelFnt = new Font("Times New Roman", Font.
    PLAIN, 16);
41      JLabel lblProjectTasks = new JLabel();
42      JLabel lblProjectTable = new JLabel();
43      JLabel lblTeamList = new JLabel();
44      JLabel lblTeamMembers = new JLabel();
45      JLabel lblGanttChart = new JLabel();
46
47
48      public static DefaultListModel<String>
    TeamListNames;
49      public static JList<String> TeamList;
50
51      public static ArrayList<String> ProjectBoxList;
52      public static JComboBox cbxProjectList;
53      JTable ProjectTable;
54      public static Object[][] tasktable;
55      public String[] columnNames = {"Task Description"
    , "Assigned to Team", "Start", "End", "Completed", "
    Reliant on:"};
56      public GridBagConstraints gbc;
57
58      public JTextArea txtInfoArea = new JTextArea(30,
    30);
59
60      JScrollPane tableSP;
61
62      public Coursework() {
63          JVM();
64      }
65
66
67      private void JVM(){
68          load_gui();
69
70          btnNewProject = mb.makeNavigationButton( "New
     Project","NewProject",
71                  "Create a new project");
```

```
 72          toolBar.add(btnNewProject);
 73          toolBar.addSeparator();
 74
 75          btnAddTeam = mb.makeNavigationButton( "Add
    Task","AddTask",
 76                  "Create a new task");
 77          toolBar.add(btnAddTeam);
 78          toolBar.addSeparator();
 79
 80          btnCreateTeam = mb.makeNavigationButton( "
    Create Team","CrtTeam",
 81                  "Create and add a team to the
    project");
 82          toolBar.add(btnCreateTeam);
 83          toolBar.addSeparator();
 84
 85          btnAssignPerson = mb.makeNavigationButton( "
    Assign Team to Task", "AssignTeam",
 86                  "Assign a person to a team");
 87          toolBar.add(btnAssignPerson);
 88          toolBar.addSeparator();
 89
 90          btnSave = mb.makeNavigationButton( "Save","
    Save",
 91                  "Save everything");
 92          toolBar.add(btnSave);
 93          toolBar.addSeparator();
 94
 95          btncritPath = mb.makeNavigationButton( "
    Critical Path","cp",
 96                  "Display the critical path of your
    project");
 97          toolBar.add(btncritPath);
 98
 99          lblProjectTasks.setText("Projects:");
100          lblProjectTasks.setFont(LabelFnt);
101
102          lblProjectTable.setText("Project Table");
103          lblProjectTable.setFont(LabelFnt);
104
105          lblGanttChart.setText("Time till completion
     (Gantt Chart)");
106          lblGanttChart.setFont(LabelFnt);
107
```

```
108         lblTeamList.setText("List of Teams");
109         lblTeamList.setFont(LabelFnt);
110 
111         lblTeamMembers.setText("Information:");
112         lblTeamMembers.setFont(LabelFnt);
113 
114         JScrollPane listSP = new JScrollPane(
    TeamList);
115         listSP.setPreferredSize(new Dimension(200,
    200));
116 
117 
118 
119         txtInfoArea.setEditable(false);
120 
121         //Explanation for gridlayout is on yt and
    the oracle website Abidon.
122         //read and explain it afterwards
123         //https://www.youtube.com/watch?v=
    ZipG38DJJK8
124         //https://docs.oracle.com/javase/tutorial/
    uiswing/layout/grid.html
125         //https://docs.oracle.com/javase/tutorial/
    uiswing/layout/gridbag.html
126         gbc = new GridBagConstraints();
127         gbc.insets = new Insets(15,15, 15, 15);
128 
129 //left panel work
130         gbc.gridx = 0;
131         gbc.gridy = 0;
132         gbc.weighty = 0;
133         gbc.weightx = 0;
134         gbc.gridwidth = 4;
135         gbc.fill = 4;
136         westpnl.add(lblProjectTasks, gbc);
137 
138         gbc.gridx = 0;
139         gbc.gridy = 1;
140         gbc.weighty = 0;
141         gbc.weightx = 0;
142         gbc.gridwidth = 4;
143         gbc.fill = 4;
144         westpnl.add(cbxProjectList, gbc);
145 
```

```
146
147
148 //centerpanel work
149         gbc.gridx = 0;
150         gbc.gridy = 0;
151         gbc.weighty = 0;
152         gbc.weightx = 0;
153         gbc.gridwidth = 4;
154         gbc.fill = 4;
155         centerpnl.add(lblProjectTable,gbc);
156
157         gbc.gridx = 0;
158         gbc.gridy = 1;
159         gbc.weighty = 0;
160         gbc.weightx = 0;
161         gbc.gridwidth = 4;
162         gbc.fill = 4;
163         centerpnl.add(tableSP,gbc);
164
165         gbc.gridx = 0;
166         gbc.gridy = 2;
167         gbc.weighty = 0;
168         gbc.weightx = 0;
169         gbc.gridwidth = 4;
170         gbc.fill = 4;
171         centerpnl.add(lblGanttChart,gbc);
172
173         //gantt chart comes here. add a scrollpane
    to it if it contains alot of data
174      /* gbc.gridx = 0;
175         gbc.gridy = 3;
176         gbc.weighty = 0;
177         gbc.weightx = 0;
178         gbc.gridwidth = 4;
179         gbc.fill = 4;
180         centerpnl.add(_____,gbc); //Gantt Chart
    variable goes here
181 */
182
183
184 //eastpanel work
185
186         gbc.gridx = 0;
187         gbc.gridy = 0;
```

```
188         gbc.weighty = 0;
189         gbc.weightx = 0;
190         gbc.gridwidth = 4;
191         gbc.fill = 4;
192         eastpnl.add(lblTeamList,gbc);
193 
194         gbc.gridx = 0;
195         gbc.gridy = 1;
196         gbc.weighty = 0;
197         gbc.weightx = 0;
198         gbc.gridwidth = 4;
199         gbc.fill = 4;
200         eastpnl.add(listSP,gbc);
201 
202         gbc.gridx = 0;
203         gbc.gridy = 2;
204         gbc.weighty = 0;
205         gbc.weightx = 0;
206         gbc.gridwidth = 4;
207         gbc.fill = 4;
208         eastpnl.add(lblTeamMembers,gbc);
209 
210         gbc.gridx = 0;
211         gbc.gridy = 3;
212         gbc.weighty = 0;
213         gbc.weightx = 0;
214         gbc.gridwidth = 4;
215         gbc.fill = 4;
216         eastpnl.add(txtInfoArea,gbc);
217 
218         frame.add(toolBar, BorderLayout.NORTH);
219         frame.add(westpnl, BorderLayout.WEST);
220         frame.add(centerpnl, BorderLayout.CENTER);
221         frame.add(eastpnl, BorderLayout.EAST);
222 
223 
224         frame.setExtendedState(JFrame.MAXIMIZED_BOTH
    );
225         frame.setTitle("JVM Coursework");
226         frame.setDefaultCloseOperation(JFrame.
    EXIT_ON_CLOSE);
227         frame.setVisible(true);
228     }
229 
```

```
230     public void load_gui(){
231         if (!data.getProjects().isEmpty()){
232             tasktable = new Object[data.
    getCurrentProject().getProjectTasks().size()][6];
233             ArrayList<Task> t = data.
    getCurrentProject().getProjectTasks();
234             for (int i = 0; i < data.
    getCurrentProject().getProjectTasks().size(); i++){
235                 Task f = data.getCurrentProject().
    getProjectTasks().get(i);
236                 Object[] arr = {f.getTaskDesc(), f.
    getAssignedTeam().getTeamName(), f.getTaskStartDate
    (), f.getTaskEndDate(), f.getIsCompleted(), f.
    getPredecessors()};
237                 for (int j = 0; j < 6; j++){
238                     tasktable[i][j] = arr[j];
239                 }
240             }
241             ProjectBoxList = new ArrayList<>();
242             for (Project i: data.getProjects()){
243                 ProjectBoxList.add(i.getProjName());
244             }
245             cbxProjectList= new JComboBox(
    ProjectBoxList.toArray());
246             cbxProjectList.setSelectedIndex(data.
    getProjects().indexOf(data.getCurrentProject()));
247             cbxProjectList.addActionListener(this);
248             cbxProjectList.setActionCommand("
    ProjectFocusChanged");
249 
250             TeamListNames = new DefaultListModel
    <>();
251             for (Team i: data.getCurrentProject().
    getProjectTeams()){
252                 TeamListNames.addElement(i.
    getTeamName());
253             }
254             TeamList = new JList<>(TeamListNames);
255 
256             ProjectTable = new JTable(tasktable,
    columnNames);
257             ProjectTable.setCellSelectionEnabled(
    false);
258 
```

```
259             tableSP = new JScrollPane(ProjectTable);
260             tableSP.setPreferredSize(new Dimension(
    800,400));
261             centerpnl.add(tableSP,gbc);
262 
263             txtInfoArea.append(data.infoText);
264 
265             appendInfo("Now showing: " +data.
    getCurrentProject().getProjName() + "\n");
266         }else{
267             data.CreateProject("new project", "
    example project", String.valueOf(LocalDate.now()),10
    , true);
268             data.getCurrentProject().addTask("Say
    hello to world",2, "null");
269             load_gui();
270         }
271     }
272 
273     public void appendInfo(String i){
274         data.infoText.concat(i);
275         txtInfoArea.append(i);
276     }
277 
278 
279     @Override
280     public void actionPerformed(ActionEvent ae) {
281 
282         if ("cp".equals(ae.getActionCommand())){
283             appendInfo("Critical Path: " + path.
    calculate_critical_path() + "\n");
284         }
285         if ("ProjectFocusChanged".equals(ae.
    getActionCommand())){
286             JComboBox cb = (JComboBox)ae.getSource
    ();
287             String projName = (String)cb.
    getSelectedItem();
288             data.setCurrentProject(data.
    getProjOfName(projName));
289             Coursework c = new Coursework();
290             frame.setVisible(false);
291         }
292 
```

```
293         if ("CrtTeam".equals(ae.getActionCommand
    ())) {
294             CreateTeam_gui pj = new CreateTeam_gui(
    this);
295         }
296 
297         if ("NewProject".equals(ae.getActionCommand
    ())) {
298             Project_gui pj = new Project_gui(this);
299         }
300 
301         if ("AddTask".equals(ae.getActionCommand
    ())) {
302             Task_gui tk = new Task_gui(this);
303         }
304 
305         if ("AssignTeam".equals(ae.getActionCommand
    ())) {
306             AssignTeam_gui tm = new AssignTeam_gui(
    this);
307         }
308 
309         if ("Save".equals(ae.getActionCommand())) {
310             int SaveConfirmation = JOptionPane.
    showConfirmDialog(null,
311                     "Do you want to save all the
    recent changes?", "Save Program Message Box",
312                     JOptionPane.YES_NO_OPTION);
313 
314             if (SaveConfirmation == JOptionPane.
    YES_OPTION) {
315                 try {
316                     data.saveInstances();
317                 } catch (IOException e) {
318                     e.printStackTrace();
319                 }
320             }
321         }
322     }
323 }
324 
```

```
 1 import java.time.Instant;
 2 import java.time.LocalDate;
 3 import java.time.ZoneId;
 4 import java.time.ZoneOffset;
 5 import java.util.Date;
 6
 7 import javax.swing.*;
 8
 9 import org.jfree.chart.ChartFactory;
10 import org.jfree.chart.ChartPanel;
11 import org.jfree.chart.JFreeChart;
12 import org.jfree.data.category.
   IntervalCategoryDataset;
13 import org.jfree.data.gantt.Task;
14 import org.jfree.data.gantt.TaskSeries;
15 import org.jfree.data.gantt.TaskSeriesCollection;
16
17 public class ganttChart extends JFrame {
18
19
20
21     public ganttChart(String applicationTitle, String
    chartTitle) {
22         super(applicationTitle);
23
24         // based on the dataset we create the chart
25         JFreeChart chart = ChartFactory.
   createGanttChart(chartTitle, "Task", "Time",
   createDataset());
26
27         // Adding chart into a chart panel
28         ChartPanel chartPanel = new ChartPanel(chart
   );
29
30         // setting default size
31         chartPanel.setPreferredSize(new java.awt.
   Dimension(500, 270));
32
33         // add to contentPane
34         setContentPane(chartPanel);
35
36     }
37
38     private IntervalCategoryDataset createDataset() {
```

```
39
40          TaskSeriesCollection dataset = new
    TaskSeriesCollection();
41          TaskSeries expected = new TaskSeries("
    Expected Date");
42          ZoneId defaultZoneId = ZoneId.systemDefault
    ();
43          int numberOfTasks = 3;
44          Instant start = LocalDate.of(2020,2,1).
    atStartOfDay().toInstant(ZoneOffset.UTC);
45          Instant end = LocalDate.of(2020,2,25).
    atStartOfDay().toInstant(ZoneOffset.UTC);
46          String taskName = "Create GUI";
47          Date.from(LocalDate.of(2018, 9, 19).
    atStartOfDay().toInstant(ZoneOffset.UTC));
48
49          for (int i = 0; i < numberOfTasks; i++) {
50              expected.add(new Task(taskName,Date.from(
    start),
51                      Date.from(end)));
52              dataset.add(expected);
53          }
54
55          return dataset;
56
57      }
58
59
60      public static void main(String[] args) {
61          String chartTitle = "ProjectName";
62          ganttChart chart = new ganttChart("Gantt
    Chart",
63                  chartTitle);
64          chart.pack();
65          chart.setVisible(true);
66      }
67
68  }
```

```
 1 import classes.Task
 2 import java.util.*
 3
 4 class critical_path(h: programHandler) {
 5     val passer = h;
 6     lateinit var output: String;
 7     lateinit var tasks: ArrayList<Task>
 8     lateinit var remaining: ArrayList<Task?>
 9     lateinit var sp: Task
10     lateinit var ep: Task
11     fun calculate_critical_path(): String {
12         remaining = ArrayList(passer.currentProject.
   projectTasks)
13         tasks = ArrayList(passer.currentProject.
   projectTasks)
14         sp = tasks[0]
15         ep = tasks[tasks.size - 1]
16         output = "";
17         var this_node: Task? = sp
18         while (this_node != ep) {
19             var highest_cost = 0
20             var highest_cost_node: Task? = null
21             for (node in this_node!!.nextTasks) {
22                 if (node.duration >= highest_cost) {
23                     highest_cost = node.duration
24                     highest_cost_node = node
25                 }
26             }
27             output = output + this_node.taskDesc
28             remaining.remove(this_node)
29             this_node = highest_cost_node
30             output = "$output, "
31         }
32         output = output + this_node.taskDesc
33         return output
34     }
35
36     init {
37
38     }
39 }
```

```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5 import java.time.LocalDate;
 6
 7 public class Project_gui extends JFrame implements
   ActionListener {
 8
 9     Coursework main;
10
11     JFrame PJframe;
12     JPanel pjpnl;
13     JPanel btnpanel;
14     Font PJfnt;
15
16
17     JLabel lblName;
18     JLabel lblDescription;
19     JLabel lblStartDate;
20     JLabel lblDurations;
21
22     JTextField txtName;
23     JTextField txtDescription;
24     JTextField txtStartDate;
25     JTextField txtDurations;
26
27     public Project_gui(Coursework m){
28         main = m;
29         Project();
30     }
31
32
33
34     private void Project(){
35         PJframe = new JFrame("Project");
36         pjpnl = new JPanel();
37         btnpanel = new JPanel();
38         PJfnt = new Font("Times New Roman", Font.
   PLAIN, 16);
39
40         lblName = new JLabel("Name");
41         lblDescription = new JLabel("Description");
42         lblStartDate = new JLabel("Start Date");
```

```
43          lblDurations = new JLabel("Durations (Days)"
    );
44
45          txtName = new JTextField();
46          txtDescription = new JTextField();
47          txtStartDate = new JTextField();
48          txtDurations = new JTextField();
49
50          lblName.setFont(PJfnt);
51          lblDescription.setFont(PJfnt);
52          lblStartDate.setFont(PJfnt);
53          lblDurations.setFont(PJfnt);
54
55          JButton btnPJSave = new JButton("Create
    Project");
56          btnPJSave.setFont(PJfnt);
57          btnPJSave.addActionListener(this);
58          btnPJSave.setActionCommand("ok");
59
60
61          btnpanel.add(btnPJSave);
62
63          pjpnl.setLayout(new GridLayout(4,2, 15,15));
64          pjpnl.add(lblName);
65          pjpnl.add(txtName);
66          pjpnl.add(lblDescription);
67          pjpnl.add(txtDescription);
68          pjpnl.add(lblStartDate);
69          pjpnl.add(txtStartDate);
70          pjpnl.add(lblDurations);
71          pjpnl.add(txtDurations);
72
73
74          PJframe.add(btnpanel, BorderLayout.NORTH);
75          PJframe.add(pjpnl, BorderLayout.CENTER);
76
77          PJframe.setDefaultCloseOperation(JFrame.
    DISPOSE_ON_CLOSE);
78          PJframe.setExtendedState(JFrame.
    MAXIMIZED_BOTH);
79          PJframe.pack();
80          PJframe.setVisible(true);
81
82      }
```

```
83
84
85      @Override
86      public void actionPerformed(ActionEvent e) {
87          //none of these action command will work
    since it is only works with toolbar. ill try to
    solve this tomorrow morninig
88          if ("ok".equals(e.getActionCommand())){
89              main.data.CreateProject(txtName.getText
    (),txtDescription.getText(), txtStartDate.getText
    (), Integer.parseInt(txtDurations.getText()), true);
90              Coursework c = new Coursework();
91              main.frame.setVisible(false);
92          }
93          PJframe.setVisible(false);
94      }
95  }
```

```
 1 import classes.Project;
 2 import classes.Team;
 3 import classes.Task;
 4 import javax.swing.JFrame;
 5
 6 public class TestingClass {
 7
 8     public static void main(String[] args){
 9
10         Project proj1 = new Project("Project JVM", "
   The JVM Coursework", "2021-11-05",100);
11         proj1.createTeam("Team A","a a a a a a",01);
12         proj1.createTeam("Team B","b b b b b b b", 02
   );
13         proj1.addTask("A",30, "null");
14         proj1.addTask("B", 60, "A");
15         proj1.addTask("C", 30, "A");
16         proj1.addTask("D",20,"A");
17         proj1.addTask("E", 365,"D,B,C");
18         proj1.addTask("F",0,"E");
19         proj1.AssignTeamToTask("Team A","A");
20         proj1.AssignTeamToTask("Team A","B");
21         proj1.AssignTeamToTask("Team A","C");
22         proj1.AssignTeamToTask("Team B","D");
23         proj1.AssignTeamToTask("Team B","E");
24         proj1.AssignTeamToTask("Team B","F");
25         critical_path_improved path = new
   critical_path_improved();
26         System.out.println("==================END
   TEST==================");
27     }
28 }
29
```

```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5
 6 public class AssignTeam_gui extends JFrame implements
    ActionListener {
 7     Coursework main;
 8
 9     JFrame ATframe = new JFrame("Team Assignment");
10     JPanel tmpnl = new JPanel();
11     JPanel btnpnl = new JPanel();
12
13     Font tmfnt = new Font("Times New Roman", Font.
   PLAIN, 18);
14
15     JButton btnTMSave = new JButton("Assign");
16
17     JLabel lblTeamName = new JLabel("Team Name");
18     JLabel lblTaskName = new JLabel("Task Description
   ");
19
20     JTextField txtTeamName = new JTextField();
21     JTextArea txtTaskName = new JTextArea();
22
23     // public static void main(String[] args) {
24     //     Team tm = new Team();
25     // }
26
27     public AssignTeam_gui(Coursework m){
28         Team();
29         main = m;
30     }
31
32     private void Team(){
33         lblTeamName.setFont(tmfnt);
34         lblTaskName.setFont(tmfnt);
35         txtTeamName.setFont(tmfnt);
36         txtTaskName.setFont(tmfnt);
37
38         btnTMSave.setFont(tmfnt);
39         btnTMSave.addActionListener(this);
40         btnTMSave.setActionCommand("ok");
41
```

```
42          btnpnl.add(btnTMSave);
43
44          tmpnl.setLayout(new GridLayout(2,2, 15,15));
45          tmpnl.add(lblTeamName);
46          tmpnl.add(txtTeamName);
47          tmpnl.add(lblTaskName);
48          tmpnl.add(txtTaskName);
49
50          ATframe.add(btnpnl, BorderLayout.NORTH);
51          ATframe.add(tmpnl, BorderLayout.CENTER);
52
53          ATframe.setDefaultCloseOperation(JFrame.
    DISPOSE_ON_CLOSE);
54          ATframe.setExtendedState(JFrame.
    MAXIMIZED_BOTH);
55          ATframe.pack();
56          ATframe.setVisible(true);
57
58      }
59
60
61      @Override
62      public void actionPerformed(ActionEvent e) {
63          if ("ok".equals(e.getActionCommand())){
64              main.data.getCurrentProject().
    AssignTeamToTask(txtTeamName.getText(), txtTaskName.
    getText());
65              Coursework c = new Coursework();
66              main.frame.setVisible(false);
67          }
68          ATframe.setVisible(false);
69      }
70  }
71
```

```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5
 6 public class CreateTeam_gui extends JFrame implements
    ActionListener {
 7     JFrame CTframe = new JFrame("Create Team");
 8     JPanel tkpnl = new JPanel();
 9     JPanel btnpnl = new JPanel();
10
11     Font tkfnt = new Font("Times New Roman", Font.
   PLAIN, 18);
12
13     JButton btnTKSave = new JButton("Create Team");
14
15
16     JLabel lblName = new JLabel("Team Name");
17     JLabel lblDescription = new JLabel("Team
   description");
18     JLabel lblID = new JLabel("ID for team");
19
20     JTextField txtName = new JTextField();
21     JTextField txtDescription = new JTextField();
22     JTextField txtID = new JTextField();
23     Coursework main;
24
25     // public static void main(String[] args) {
26     //     Task tk = new Task();
27     // }
28
29     public CreateTeam_gui(Coursework m){
30         Task();
31         main = m;
32     }
33
34     private void Task(){
35         lblName.setFont(tkfnt);
36         lblDescription.setFont(tkfnt);
37         lblID.setFont(tkfnt);
38
39         btnTKSave.setFont(tkfnt);
40         btnTKSave.addActionListener(this);
41         btnTKSave.setActionCommand("ok");
```

```
42          btnpnl.add(btnTKSave);
43
44          tkpnl.setLayout(new GridLayout(3,2, 15,15));
45          tkpnl.add(lblName);
46          tkpnl.add(txtName);
47          tkpnl.add(lblDescription);
48          tkpnl.add(txtDescription);
49          tkpnl.add(lblID);
50          tkpnl.add(txtID);
51
52          CTframe.add(btnpnl, BorderLayout.CENTER);
53          CTframe.add(tkpnl, BorderLayout.NORTH);
54
55          CTframe.setDefaultCloseOperation(JFrame.
    DISPOSE_ON_CLOSE);
56          CTframe.setExtendedState(JFrame.
    MAXIMIZED_BOTH);
57          CTframe.pack();
58          CTframe.setVisible(true);
59
60      }
61
62
63      @Override
64      public void actionPerformed(ActionEvent e) {
65          if ("ok".equals(e.getActionCommand())){
66              main.data.getCurrentProject().createTeam(
    txtName.getText(), txtDescription.getText() ,Integer.
    parseInt(txtID.getText()));
67              Coursework c = new Coursework();
68              main.frame.setVisible(false);
69          }
70          CTframe.setVisible(false);
71      }
72  }
73
```

```
 1 import classes.Project;
 2 import classes.Task;
 3 import classes.Team;
 4
 5 import java.io.*;
 6 import java.time.LocalDate;
 7 import java.util.ArrayList;
 8 import java.util.Arrays;
 9
10 public class programHandler {
11
12     public static programHandler instance = new
   programHandler();
13     private static ArrayList<Project> projects;
14     private static Project currentProject;
15
16     public static String infoText;
17     public static void main(String args[]) throws
   IOException {
18         projects = new ArrayList<>();
19         infoText = "";
20         loadInstances();
21         Coursework cw = new Coursework();
22         //Task[] Sorted = path.criticalPath();
23         //for (int i = 0; i < Sorted.length; i++)
24         //    System.out.println(Sorted[i].
   getTaskDesc());
25     }
26
27     public static programHandler getInstance( ) {
28         return instance;
29     }
30
31     public void CreateProject(String name, String
   note, String startdate, int dur, boolean isCreate){
32         if (isCreate && startdate.compareTo(String.
   valueOf(LocalDate.now())) < 0){
33             System.out.println("ERROR: starting date
   is earlier than current date!");
34         } else{
35             Project tmp = new Project(name, note,
   startdate, dur);
36             projects.add(tmp);
37             currentProject = tmp;
```

```
38          }
39      }
40
41      public Project getProjOfName(String name){
42          Project a = null;
43          for (Project project : projects) {
44              if (project.getProjName().equals(name)) {
45                  a = project;
46                  break;
47              }
48          }
49          return a;
50      }
51
52      public Project getCurrentProject(){
53          return currentProject;
54      }
55
56      public void setCurrentProject(Project p){
57          currentProject  = p;
58      }
59
60      public ArrayList<Project> getProjects() { return
    projects;}
61
62      public void saveInstances() throws IOException {
63          FileWriter csvWriter = new FileWriter("
    Projects.csv");
64          for (Project j : projects) {
65              ArrayList<String> Proj_data = new
    ArrayList<>(Arrays.asList("p", j.getProjName(), j.
    getProjNote(), j.getStartDate().toString(),
66                      String.valueOf(j.getProjDuration
    ())));
67              Proj_data.forEach((c)->{
68                  try {
69                      csvWriter.append(c).append("%");
70                  } catch (IOException e) {
71                      e.printStackTrace();
72                  }
73              });
74              csvWriter.append("\n");
75              for (int h = 0; h < j.getProjectTeams().
    size(); h++) {
```

```
 76                  Team t = j.getProjectTeams().get(h);
 77                  ArrayList<String> team_data = new
    ArrayList<> (Arrays.asList("g", t.getTeamName(), t.
    getTeamDescription(), String.valueOf(t.getID())));
 78                  team_data.forEach((c)->{
 79                      try {
 80                          csvWriter.append(c).append(
    "%");
 81                      } catch (IOException e) {
 82                          e.printStackTrace();
 83                      }
 84                  });
 85                  csvWriter.append("\n");
 86              }
 87              for (int h = 0; h < j.getProjectTasks().
    size(); h++) {
 88                  Task t = j.getProjectTasks().get(h);
 89                  ArrayList<String> task_data = new
    ArrayList<>(Arrays.asList("t", t.getTaskDesc(), t.
    getAssignedProj().getProjName(),
 90                          String.valueOf(t.getDuration
    ()), t.getAssignedTeam().getTeamName(), t.
    getPredecessors()));
 91                  task_data.forEach((c)->{
 92                      try {
 93                          csvWriter.append(c).append(
    "%");
 94                      } catch (IOException e) {
 95                          e.printStackTrace();
 96                      }
 97                  });
 98                  csvWriter.append("\n");
 99              }
100          }
101          csvWriter.flush();
102          csvWriter.close();
103      }
104
105      public static void loadInstances() throws
    IOException {
106          File csvFile = new File("Projects.csv");
107          if (csvFile.isFile()) {
108              BufferedReader csvReader = new
    BufferedReader(new FileReader("Projects.csv"));
```

```
109             String row;
110             while ((row = csvReader.readLine()) !=
    null) {
111                 String[] data = row.split("%");
112                 if (data[0].equals("p")){
113                     instance.CreateProject(data[1],
    data[2],data[3],Integer.parseInt(data[4]),false);
114                 }
115                 if (data[0].equals("g")){
116                     instance.currentProject.
    createTeam(data[1], data[2], Integer.parseInt(data[3
    ]));
117                 }
118                 if(data[0].equals("t")){
119                     instance.currentProject.addTask(
    data[1],Integer.parseInt(data[3]),data[5]);
120                     if (!data[4].equals("null")){
121                         instance.currentProject.
    AssignTeamToTask(data[4],data[1]);
122                     }
123                 }
124             }
125             csvReader.close();
126         }
127
128
129     }
130
131 }
132
```

```
 1 import classes.Task;
 2
 3 import java.util.*;
 4
 5 //Code is based from that of: https://stackoverflow.
   com/questions/2985317/critical-path-method-algorithm
 6 //Adapted to use my classes
 7
 8 public class critical_path_improved {
 9
10     public static programHandler handler =
   programHandler.getInstance();
11
12     public static Task[] criticalPath(){
13         //tasks whose critical cost has been
   calculated
14         ArrayList<Task> completed = new ArrayList<
   Task>();
15         //tasks whose ciritcal cost needs to be
   calculated
16         ArrayList<Task> remaining = new ArrayList<
   Task>();
17         for (int i = handler.getCurrentProject().
   getProjectTasks().size() -1; i >= 0; i--){
18             remaining.add(handler.getCurrentProject
   ().getProjectTasks().get(i));
19         }
20
21         //Backflow algorithm
22         //while there are tasks whose critical cost
   isn't calculated.
23         while(!remaining.isEmpty()){
24             boolean progress = false;
25
26             //find a new task to calculate
27             for(Iterator<Task> it = remaining.
   iterator(); it.hasNext();){
28                 Task task = it.next();
29                 if(completed.containsAll(task.
   getNextTasks())){
30                     //all dependencies calculated,
   critical cost is max dependency
31                     //critical cost, plus our cost
32                     int critical = 0;
```

```
33                      for(Task t : task.getNextTasks
   ()){
34                          if(t.criticalCost > critical
   ){
35                              critical = t.criticalCost
   ;
36                          }
37                      }
38                      task.criticalCost = critical+task
   .getDuration();
39                      System.out.println(task.
   criticalCost);
40                      //set task as calculated an
   remove
41                      completed.add(task);
42                      it.remove();
43                      //note we are making progress
44                      progress = true;
45                  }
46              }
47              //If we haven't made any progress then a
   cycle must exist in
48              //the graph and we wont be able to
   calculate the critical path
49              if(!progress) throw new RuntimeException(
   "Cyclic dependency, algorithm stopped!");
50          }
51
52          //get the tasks
53          Task[] ret = completed.toArray(new Task[0]);
54          //create a priority list
55          Arrays.sort(ret, (o1, o2) -> {
56              //sort by cost
57              int i= o2.criticalCost-o1.criticalCost;
58              if(i != 0)return i;
59
60              //using dependency as a tie breaker
61              //note if a is dependent on b then
62              //critical cost a must be >= critical
   cost of b
63              if(o1.isDependant(o2))return -1;
64              if(o2.isDependant(o1))return 1;
65              return 0;
66          });
```

```
67          return ret;
68      }
69  }
70
```