

```

01 package playable;
02
03 import java.awt.Dimension;
04
05 import javax.swing.*;
06
07 import java.net.URL;
08 import java.io.IOException;
09
10 /**
11  * @author Jory Stewart
12  * The purpose of this class is to create a separate frame displaying
13  * some basic information regarding the program.
14  */
15 public class About {
16     static JFrame versionNotesFrame, differencesFrame, helpFrame;
17     /**
18      * Shows a opening frame displaying some basic information.
19      */
20     public static void createAndShowAbout()
21     {
22         JOptionPane.showMessageDialog(null, "AOText version 0.1a \nCopyright
23         2007 Jory Stewart \n" +
24         "Anarchy Online and Funcom are copyright Funcom.\nTo send a
25         bug report or suggestions/comments, see Contact Me under the Help menu.
26         ", "About AOText", JOptionPane.INFORMATION_MESSAGE);
27     }
28     public static void showContactInfo()
29     {
30         JOptionPane.showMessageDialog(null, "To send bug reports, sugges
31         tions, or comments, I can be reached at:\nsolidthanatos@gmail.com or so
32         lid_sephiroth@hotmail.com\nPlease include 'AOText' in the subject line,
33         or I will consider it spam.", "Contact Info", JOptionPane.INFORMATION_
34         MESSAGE);
35     }
36     public static void createAndShowVersionNotes()
37     {
38         versionNotesFrame = new JFrame("Version Notes");
39         JEditorPane editorPane = new JEditorPane();
40         editorPane.setEditable(false);
41         java.net.URL helpURL = About.class.getResource(
42             "Version Notes.html");
43         if (helpURL != null) {
44             try {
45                 editorPane.setPage(helpURL);
46             } catch (IOException e) {
47                 System.err.println("Attempted to read a bad URL: " + hel
48                 pURL);
49             }
50         } else {
51             System.err.println("Couldn't find file: Version Notes.html")
52         }
53     }
54     ;
55 }
56 // Put the editor pane in a scroll pane.

```

```

47     JScrollPane editorScrollPane = new JScrollPane(editorPane);
48     editorScrollPane.setVerticalScrollBarPolicy(
49         JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
50     editorScrollPane.setPreferredSize(new Dimension(250, 145));
51     editorScrollPane.setMinimumSize(new Dimension(10, 10));
52     versionNotesFrame.getContentPane().add(editorScrollPane);
53
54     versionNotesFrame.setBounds(25, 25, 350, 350);
55     versionNotesFrame.setLocationRelativeTo(null);
56     versionNotesFrame.pack();
57     versionNotesFrame.setVisible(true);
58 }
59 public static void createAndShowDifferences()
60 {
61     differencesFrame = new JFrame("AO/AOText Differences");
62     JEditorPane editorPane = new JEditorPane();
63     editorPane.setEditable(false);
64     java.net.URL helpURL = About.class.getResource(
65         "Differences.html");
66     if (helpURL != null) {
67         try {
68             editorPane.setPage(helpURL);
69         } catch (IOException e) {
70             System.err.println("Attempted to read a bad URL: " + hel
pURL);
71         }
72     } else {
73         System.err.println("Couldn't find file: Differences.html");
74     }
75
76 //     Put the editor pane in a scroll pane.
77     JScrollPane editorScrollPane = new JScrollPane(editorPane);
78     editorScrollPane.setVerticalScrollBarPolicy(
79         JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
80     editorScrollPane.setPreferredSize(new Dimension(300, 250));
81     editorScrollPane.setMinimumSize(new Dimension(10, 10));
82     differencesFrame.getContentPane().add(editorScrollPane);
83
84     differencesFrame.setBounds(25, 25, 350, 350);
85     differencesFrame.setLocationRelativeTo(null);
86     differencesFrame.pack();
87     differencesFrame.setVisible(true);
88 }
89 public static void createAndShowHelp()
90 {
91     helpFrame = new JFrame("Help");
92     helpFrame.setBounds(25, 25, 350, 350);
93     helpFrame.setLocationRelativeTo(null);
94     helpFrame.pack();
95     helpFrame.setVisible(true);
96 }
97 }

```

```

001 package playable;
002
003 import java.awt.event.ActionEvent;
004 import java.awt.event.ActionListener;
005 import java.awt.event.KeyEvent;
006 import java.awt.*;
007
008 import javax.swing.*;
009 import javax.swing.text.*;
010
011 import java.io.*;
012
013 public class CreateChar implements ActionListener{
014
015     JMenuItem exit, saveChar, exit2, saveChar2;
016     JMenuBar menuBar, oppMenuBar;
017     JPanel mainPanel;
018     JFormattedTextField nameEntry;
019     String charName, breed, gender, prof;
020     static JFrame charFrame, oppFrame;
021     final JFileChooser fc = new JFileChooser();
022
023     public JMenuBar createCharMenuBar()
024     {
025         menuBar = new JMenuBar();
026         JMenu file = new JMenu("File");
027         file.setMnemonic(KeyEvent.VK_F);
028         saveChar = new JMenuItem("Save Character");
029         saveChar.setMnemonic(KeyEvent.VK_S);
030         saveChar.addActionListener(this);
031         exit = new JMenuItem("Exit");
032         exit.setMnemonic(KeyEvent.VK_F4);
033         exit.addActionListener(this);
034         file.add(saveChar);
035         file.addSeparator();
036         file.add(exit);
037         menuBar.add(file);
038         return menuBar;
039     }
040     public JMenuBar createOppMenuBar()
041     {
042         oppMenuBar = new JMenuBar();
043         JMenu file2 = new JMenu("File");
044         file2.setMnemonic(KeyEvent.VK_F);
045         saveChar2 = new JMenuItem("Save Opponent");
046         saveChar2.setMnemonic(KeyEvent.VK_S);
047         saveChar2.addActionListener(this);
048         exit2 = new JMenuItem("Exit");
049         exit2.setMnemonic(KeyEvent.VK_F4);
050         exit2.addActionListener(this);
051         file2.add(saveChar2);
052         file2.addSeparator();
053         file2.add(exit2);
054         oppMenuBar.add(file2);
055         return oppMenuBar;

```

```

056 }
057 public JPanel createMainPanel()
058 {
059     mainPanel = new JPanel(new GridBagLayout());
060     GridBagConstraints c = new GridBagConstraints();
061     JLabel nameLabel = new JLabel("Name:");
062     JLabel breedLabel = new JLabel("Breed:");
063     JLabel genderLabel = new JLabel("Gender:");
064     JLabel profLabel = new JLabel("Profession:");
065     JLabel weaponLabel = new JLabel("Main Weapon:");
066     nameEntry = new JFormattedTextField(createFormatter("ULLLLLLLLLLL
LLL"));
067     nameEntry.setColumns(13);
068     String[] breedList = {"Solitus", "Opifex", "Nanomage", "Atrox"}
;
069     JComboBox breedSelection = new JComboBox(breedList);
070     JRadioButton male = new JRadioButton("Male");
071     male.addActionListener(this);
072     JRadioButton female = new JRadioButton("Female");
073     female.addActionListener(this);
074     ButtonGroup genderSel = new ButtonGroup();
075     genderSel.add(male);
076     genderSel.add(female);
077     String[] profList = {"Adventurer", "Agent", "Bureaucrat", "Doct
or", "Enforcer", "Engineer", "Fixer", "Keeper", "Martial Artist", "Meta
-Physicist", "Nano-Technician", "Shade", "Soldier", "Trader"};
078     JComboBox profSelection = new JComboBox(profList);
079     //if (profSelection == "Nano-Technician") {}
080     String[] weaponsList = {"Ofab Mongoose Mk I [1HE]", "Ofab Wolf
Mk I [2HE]", "Ofab Panther Mk I [1HB]", "Ofab Bear Mk I [2HB]", "Ofab V
iper Mk I [Piercing]", "Dreadloch Shen Sticks [MA]", "Dreadlock Reaper
[Melee Energy]", "Ofab Peregrine Mk I [Pistol]", "Ofab Shark Mk I [Assa
ult Rifle]", "Ofab Silverback Mk I [Shotgun]", "Ofab Cobra Mk I [Rifle]
", "Ofab Hawk Mk I [SMG]", "Ofab Tiger Mk I [Bow]", "Dreadloch Lancer [
Ranged Energy]", "Dreadloch Cannon [Heavy Weapons]", "Dreadloch Shruike
n [Sharp Objects]", "Dreadloch Redliner [Grenade]", "Basic Cyberdeck [Na
no-Technician only, no attack skill]"};
081     JComboBox weaponSelection = new JComboBox(weaponsList);
082     c.anchor = GridBagConstraints.LINE_START;
083     c.gridx = 0;
084     c.gridy = 0;
085     mainPanel.add(nameLabel, c);
086     c.gridx = 1;
087     mainPanel.add(nameEntry, c);
088     c.gridx = 0;
089     c.gridy = 1;
090     mainPanel.add(breedLabel, c);
091     c.gridx = 1;
092     mainPanel.add(breedSelection, c);
093     c.gridx = 0;
094     c.gridy = 2;
095     mainPanel.add(genderLabel, c);
096     c.gridx = 1;
097     mainPanel.add(male, c);
098     c.gridx = 2;
099     mainPanel.add(female, c);

```

```

100     c.gridx = 0;
101     c.gridy = 3;
102     mainPanel.add(profLabel, c);
103     c.gridx = 1;
104     mainPanel.add(profSelection, c);
105     c.gridx = 0;
106     c.gridy = 4;
107     mainPanel.add(weaponLabel, c);
108     c.gridx = 1;
109     c.gridwidth = 2;
110     mainPanel.add(weaponSelection, c);
111     return mainPanel;
112 }
113 protected MaskFormatter createFormatter(String s) {
114     MaskFormatter formatter = null;
115     try {
116         formatter = new MaskFormatter(s);
117     } catch (java.text.ParseException exc) {
118         System.err.println("formatter is bad: " + exc.getMessage(
119 ));
119         System.exit(-1);
120     }
121     return formatter;
122 }
123 public void actionPerformed(ActionEvent e)
124 {
125     if (e.getSource() == exit) {charFrame.setVisible(false);}
126     else if (e.getSource() == exit2) {oppFrame.setVisible(false);}
127     else if (e.getSource() == saveChar || e.getSource() == saveChar
128 2)
129     {
130         int returnVal = fc.showSaveDialog(charFrame);
131         if (returnVal == JFileChooser.APPROVE_OPTION) {
132             File file = fc.getSelectedFile();
133             //boolean exists = file.exists();
134             //if (exists == false) {file.createNewFile();}
135         }
136     }
137 public static void createAndShowCharCreation()
138 {
139     charFrame = new JFrame("Character Creation");
140     charFrame.setIconImage(new ImageIcon("Images/mainLogo.png").get
141 Image());
142     CreateChar test = new CreateChar();
143     JPanel contentPane = new JPanel();
144     charFrame.setJMenuBar(test.createCharMenuBar());
145     charFrame.setContentPane(contentPane);
146     contentPane.add(test.createMainPanel());
147
148     charFrame.setBounds(25, 25, 750, 750);
149     charFrame.setLocationRelativeTo(null);
150     charFrame.pack();
151     charFrame.setVisible(true);
152 }
153 public static void createAndShowOppCreation()

```

```
153 {
154     oppFrame = new JFrame("Customize Opponent");
155     oppFrame.setIconImage(new ImageIcon("Images/mainLogo.png").getI
mage());
156     CreateChar test2 = new CreateChar();
157     JPanel contentPane2 = new JPanel();
158     oppFrame.setJMenuBar(test2.createOppMenuBar());
159     oppFrame.setContentPane(contentPane2);
160     contentPane2.add(test2.createMainPanel());
161
162     oppFrame.setBounds(25, 25, 750, 750);
163     oppFrame.setLocationRelativeTo(null);
164     oppFrame.pack();
165     oppFrame.setVisible(true);
166 }
167 }
```

[Java2html](#)

```

001 package playable;
002
003 import java.awt.*;
004 import java.awt.event.*;
005 import javax.swing.event.*;
006
007 import javax.swing.*;
008
009 public class IPWindow implements ActionListener, ChangeListener{
010
011     JPanel mainAtt, body, melee, ranged, otherWeaps, nano, speed, IPInfo;
012     JMenuBar menuBar;
013     JMenu file;
014     JLabel remainingIP;
015     JMenuItem saveChar, exit, resetIP;
016     Integer str, agi, stam, intel, sen, psy;
017     Integer bodyDev, nanoPool;
018     Integer oneHE, twoHE, fastAtk, brawl, sneakAttack, dimach, parry,
        riposte, multiMelee, MA, oneHB, twoHB, piercing, meleeEner;
019     Integer pistol, rifle, bow, smg, shotgun, AR, RE, flingShot, burst,
        FA, AS, multiRanged;
020     Integer sharpObj, grenade, heavyWeaps;
021     Integer matCrea, matMet, bioMet, timeSpace, senseImp, psyMod, firstAid;
022     Integer evadeCls, dodgeRng, duckExp, physInit, meleeInit, rangedInit,
        nanoInit, nanoRes;
023     JSpinner strSpinner, agiSpinner, intelSpinner, stamSpinner, senSpinner,
        psySpinner;
024     JSpinner bDevSpinner, nPoolSpinner;
025     JSpinner oneHESpinner, twoHESpinner, fastAttackSpinner, brawlSpinner,
        sneakAttackSpinner, dimachSpinner, parrySpinner, riposteSpinner,
        multiMeleeSpinner, MASpinner, oneHBSpinner, twoHBSpinner, piercingSpinner,
        meleeEnerSpinner;
026     JSpinner pistolSpinner, rifleSpinner, bowSpinner, mgsmgSpinner,
        shotgunSpinner, assaultRifleSpinner, rangedEnerSpinner, flingShotSpinner,
        burstSpinner, fullAutoSpinner, aimedShotSpinner, multiRangedSpinner;
027     JSpinner sharpObjSpinner, grenadeSpinner, heavyWeapsSpinner;
028     JSpinner matCreaSpinner, matMetSpinner, bioMetSpinner, timeSpaceSpinner,
        senseImpSpinner, psyModSpinner, firstAidSpinner;
029     JSpinner evadeClsSpinner, dodgeRngSpinner, duckExpSpinner, physInitSpinner,
        meleeInitSpinner, rangedInitSpinner, nanoInitSpinner, nanoResSpinner;
030     Integer IP;
031     //Testing purposes
032     SpinnerModel strSpinnerModel;
033
034
035     static JFrame ipWindow;
036
037     public JMenuBar createMenuBar()
038     {
039         menuBar = new JMenuBar();
040         file = new JMenu("File");
041         saveChar = new JMenuItem("Save IP Allocation");

```

```

042     saveChar.addActionListener(this);
043     resetIP = new JMenuItem("Reset IP");
044     resetIP.addActionListener(this);
045     exit = new JMenuItem("Exit");
046     exit.addActionListener(this);
047     file.add(saveChar);
048     file.add(resetIP);
049     file.addSeparator();
050     file.add(exit);
051     menuBar.add(file);
052     return menuBar;
053 }
054 public JPanel createMainAttPane()
055 {
056     mainAtt = new JPanel(new GridBagLayout());
057     GridBagConstraints c = new GridBagConstraints();
058     JLabel strLabel = new JLabel("Strength:");
059     JLabel agiLabel = new JLabel("Agility:");
060     JLabel staLabel = new JLabel("Stamina:");
061     JLabel intelLabel = new JLabel("Intelligence:");
062     JLabel senLabel = new JLabel("Sense:");
063     JLabel psyLabel = new JLabel("Psychic:");
064     SpinnerModel strModel = new SpinnerNumberModel(0, 0, 5000, 1);
065     strModel.addChangeListener(this);
066     SpinnerModel agiModel = new SpinnerNumberModel(0, 0, 5000, 1);
067     SpinnerModel staModel = new SpinnerNumberModel(0, 0, 5000, 1);
068     SpinnerModel intelModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
069     SpinnerModel senModel = new SpinnerNumberModel(0, 0, 5000, 1);
070     SpinnerModel psyModel = new SpinnerNumberModel(0, 0, 5000, 1);
071     JSpinner strSpinner = new JSpinner(strModel);
072     JSpinner agiSpinner = new JSpinner(agiModel);
073     JSpinner staSpinner = new JSpinner(staModel);
074     JSpinner intelSpinner = new JSpinner(intelModel);
075     JSpinner senSpinner = new JSpinner(senModel);
076     JSpinner psySpinner = new JSpinner(psyModel);
077     c.gridx = 1;
078     c.gridy = 0;
079     c.anchor = GridBagConstraints.PAGE_START;
080     mainAtt.add(strLabel, c);
081     c.gridx = 2;
082     mainAtt.add(strSpinner, c);
083     c.gridx = 1;
084     c.gridy = 1;
085     mainAtt.add(agiLabel, c);
086     c.gridx = 2;
087     mainAtt.add(agiSpinner, c);
088     c.gridx = 1;
089     c.gridy = 2;
090     mainAtt.add(staLabel, c);
091     c.gridx = 2;
092     mainAtt.add(staSpinner, c);
093     c.gridx = 1;
094     c.gridy = 3;
095     mainAtt.add(intelLabel, c);
096     c.gridx = 2;

```

```

097     mainAtt.add(intelSpinner, c);
098     c.gridx = 1;
099     c.gridy = 4;
100     mainAtt.add(senLabel, c);
101     c.gridx = 2;
102     mainAtt.add(senSpinner, c);
103     c.gridx = 1;
104     c.gridy = 5;
105     mainAtt.add(psyLabel, c);
106     c.gridx = 2;
107     mainAtt.add(psySpinner, c);
108     return mainAtt;
109 }
110 public JPanel createBodyPane()
111 {
112     SpinnerModel bDevModel = new SpinnerNumberModel(0, 0, 5000, 1);
113     SpinnerModel nPoolModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
114     JSpinner bDevSpinner = new JSpinner(bDevModel);
115     JSpinner nPoolSpinner = new JSpinner(nPoolModel);
116     body = new JPanel(new GridBagLayout());
117     GridBagConstraints c = new GridBagConstraints();
118     JLabel bDevLabel = new JLabel("Body Development:");
119     JLabel nPoolLabel = new JLabel("Nano Pool:");
120     c.gridx = 1;
121     c.gridy = 0;
122     body.add(bDevLabel, c);
123     c.gridx = 2;
124     body.add(bDevSpinner, c);
125     c.gridx = 1;
126     c.gridy = 1;
127     body.add(nPoolLabel, c);
128     c.gridx = 2;
129     body.add(nPoolSpinner, c);
130     return body;
131 }
132 public JPanel createMeleePane()
133 {
134     melee = new JPanel(new GridBagLayout());
135     GridBagConstraints c = new GridBagConstraints();
136     JLabel oneHELabel = new JLabel("1 Hand Edged Weapons:");
137     JLabel twoHELabel = new JLabel("2 Hand Edged Weapons:");
138     JLabel fastAttackLabel = new JLabel("Fast Attack:");
139     JLabel brawlLabel = new JLabel("Brawl:");
140     JLabel sneakAttackLabel = new JLabel("Sneak Attack:");
141     JLabel dimachLabel = new JLabel("Dimach:");
142     JLabel parryLabel = new JLabel("Parry:");
143     JLabel riposteLabel = new JLabel("Riposte:");
144     JLabel multiMeleeLabel = new JLabel("Multiple Melee Weapons:");
145     JLabel MALabel = new JLabel("Martial Arts:");
146     JLabel oneHBLLabel = new JLabel("1 Hand Blunt Weapons:");
147     JLabel twoHBLLabel = new JLabel("2 Hand Blunt Weapons:");
148     JLabel piercingLabel = new JLabel("Piercing:");
149     JLabel meleeEnerLabel = new JLabel("Melee Energy Weapons:");
150     SpinnerModel oneHEModel = new SpinnerNumberModel(0, 0, 5000, 1)
;

```

```

151     SpinnerModel twoHEModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
152     SpinnerModel fastAttackModel = new SpinnerNumberModel(0, 0, 500
0, 1);
153     SpinnerModel brawlModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
154     SpinnerModel sneakAttackModel = new SpinnerNumberModel(0, 0, 50
00, 1);
155     SpinnerModel dimachModel = new SpinnerNumberModel(0, 0, 5000, 1
);
156     SpinnerModel parryModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
157     SpinnerModel riposteModel = new SpinnerNumberModel(0, 0, 5000,
1);
158     SpinnerModel multiMeleeModel = new SpinnerNumberModel(0, 0, 500
0, 1);
159     SpinnerModel MAModel = new SpinnerNumberModel(0, 0, 5000, 1);
160     SpinnerModel oneHBModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
161     SpinnerModel twoHBModel = new SpinnerNumberModel(0, 0, 5000, 1)
;
162     SpinnerModel piercingModel = new SpinnerNumberModel(0, 0, 5000,
1);
163     SpinnerModel meleeEnerModel = new SpinnerNumberModel(0, 0, 5000
, 1);
164     JSpinner oneHESpinner = new JSpinner(oneHEModel);
165     JSpinner twoHESpinner = new JSpinner(twoHEModel);
166     JSpinner fastAttackSpinner = new JSpinner(fastAttackModel);
167     JSpinner brawlSpinner = new JSpinner(brawlModel);
168     JSpinner sneakAttackSpinner = new JSpinner(sneakAttackModel);
169     JSpinner dimachSpinner = new JSpinner(dimachModel);
170     JSpinner parrySpinner = new JSpinner(parryModel);
171     JSpinner riposteSpinner = new JSpinner(riposteModel);
172     JSpinner multiMeleeSpinner = new JSpinner(multiMeleeModel);
173     JSpinner MASpinner = new JSpinner(MAModel);
174     JSpinner oneHBSpinner = new JSpinner(oneHBModel);
175     JSpinner twoHBSpinner = new JSpinner(twoHBModel);
176     JSpinner piercingSpinner = new JSpinner(piercingModel);
177     JSpinner meleeEnerSpinner = new JSpinner(meleeEnerModel);
178     c.gridx = 1;
179     c.gridy = 0;
180     melee.add(oneHELabel, c);
181     c.gridx = 2;
182     melee.add(oneHESpinner, c);
183     c.gridx = 1;
184     c.gridy = 1;
185     melee.add(twoHELabel, c);
186     c.gridx = 2;
187     melee.add(twoHESpinner, c);
188     c.gridx = 1;
189     c.gridy = 2;
190     melee.add(fastAttackLabel, c);
191     c.gridx = 2;
192     melee.add(fastAttackSpinner, c);
193     c.gridx = 1;
194     c.gridy = 3;

```

```
195     melee.add(brawlLabel, c);
196     c.gridx = 2;
197     melee.add(brawlSpinner, c);
198     c.gridx = 1;
199     c.gridy = 4;
200     melee.add(sneakAttackLabel, c);
201     c.gridx = 2;
202     melee.add(sneakAttackSpinner, c);
203     c.gridx = 1;
204     c.gridy = 5;
205     melee.add(dimachLabel, c);
206     c.gridx = 2;
207     melee.add(dimachSpinner, c);
208     c.gridx = 1;
209     c.gridy = 6;
210     melee.add(parryLabel, c);
211     c.gridx = 2;
212     melee.add(parrySpinner, c);
213     c.gridx = 1;
214     c.gridy = 7;
215     melee.add(riposteLabel, c);
216     c.gridx = 2;
217     melee.add(riposteSpinner, c);
218     c.gridx = 1;
219     c.gridy = 8;
220     melee.add(multiMeleeLabel, c);
221     c.gridx = 2;
222     melee.add(multiMeleeSpinner, c);
223     c.gridx = 1;
224     c.gridy = 9;
225     melee.add(MALabel, c);
226     c.gridx = 2;
227     melee.add(MASpinner, c);
228     c.gridx = 1;
229     c.gridy = 10;
230     melee.add(oneHBLLabel, c);
231     c.gridx = 2;
232     melee.add(oneHBSpinner, c);
233     c.gridx = 1;
234     c.gridy = 11;
235     melee.add(twoHBLLabel, c);
236     c.gridx = 2;
237     melee.add(twoHBSpinner, c);
238     c.gridx = 1;
239     c.gridy = 12;
240     melee.add(piercingLabel, c);
241     c.gridx = 2;
242     melee.add(piercingSpinner, c);
243     c.gridx = 1;
244     c.gridy = 13;
245     melee.add(meleeEnerLabel, c);
246     c.gridx = 2;
247     melee.add(meleeEnerSpinner, c);
248     c.gridx = 1;
249     return melee;
250 }
```

```

251 public JPanel createRangedPane()
252 {
253     ranged = new JPanel(new GridBagLayout());
254     GridBagConstraints c = new GridBagConstraints();
255     JLabel pistolLabel = new JLabel("Pistol:");
256     JLabel rifleLabel = new JLabel("Rifle:");
257     JLabel bowLabel = new JLabel("Bow:");
258     JLabel SMGLLabel = new JLabel("MG/SMG:");
259     JLabel shotgunLabel = new JLabel("Shotgun:");
260     JLabel assaultRifleLabel = new JLabel("Assault Rifle:");
261     JLabel rangedEnerLabel = new JLabel("Ranged Energy Weapons:");
262     JLabel flingShotLabel = new JLabel("Fling Shot:");
263     JLabel burstLabel = new JLabel("Burst:");
264     JLabel fullAutoLabel = new JLabel("Full Auto:");
265     JLabel aimedShotLabel = new JLabel("Aimed Shot:");
266     JLabel multiRangedLabel = new JLabel("Multiple Ranged Weapons:");
267     SpinnerModel pistolModel = new SpinnerNumberModel(0, 0, 5000, 1);
268     SpinnerModel rifleModel = new SpinnerNumberModel(0, 0, 5000, 1);
269     SpinnerModel bowModel = new SpinnerNumberModel(0, 0, 5000, 1);
270     SpinnerModel mgsmgModel = new SpinnerNumberModel(0, 0, 5000, 1);
271     SpinnerModel shotgunModel = new SpinnerNumberModel(0, 0, 5000, 1);
272     SpinnerModel assaultRifleModel = new SpinnerNumberModel(0, 0, 5000, 1);
273     SpinnerModel rangedEnerModel = new SpinnerNumberModel(0, 0, 500, 1);
274     SpinnerModel flingShotModel = new SpinnerNumberModel(0, 0, 5000, 1);
275     SpinnerModel burstModel = new SpinnerNumberModel(0, 0, 5000, 1);
276     SpinnerModel fullAutoModel = new SpinnerNumberModel(0, 0, 5000, 1);
277     SpinnerModel aimedShotModel = new SpinnerNumberModel(0, 0, 5000, 1);
278     SpinnerModel multiRangedModel = new SpinnerNumberModel(0, 0, 5000, 1);
279     JSpinner pistolSpinner = new JSpinner(pistolModel);
280     JSpinner rifleSpinner = new JSpinner(rifleModel);
281     JSpinner bowSpinner = new JSpinner(bowModel);
282     JSpinner mgsmgSpinner = new JSpinner(mgsmgModel);
283     JSpinner shotgunSpinner = new JSpinner(shotgunModel);
284     JSpinner assaultRifleSpinner = new JSpinner(assaultRifleModel);
285     JSpinner rangedEnerSpinner = new JSpinner(rangedEnerModel);
286     JSpinner flingShotSpinner = new JSpinner(flingShotModel);
287     JSpinner burstSpinner = new JSpinner(burstModel);
288     JSpinner fullAutoSpinner = new JSpinner(fullAutoModel);
289     JSpinner aimedShotSpinner = new JSpinner(aimedShotModel);
290     JSpinner multiRangedSpinner = new JSpinner(multiRangedModel);
291     c.gridx = 1;
292     c.gridy = 0;
293     ranged.add(pistolLabel, c);
294     c.gridx = 2;

```

```
295 ranged.add(pistolSpinner, c);
296 c.gridx = 1;
297 c.gridy = 1;
298 ranged.add(rifleLabel, c);
299 c.gridx = 2;
300 ranged.add(rifleSpinner, c);
301 c.gridx = 1;
302 c.gridy = 2;
303 ranged.add(bowLabel, c);
304 c.gridx = 2;
305 ranged.add(bowSpinner, c);
306 c.gridx = 1;
307 c.gridy = 3;
308 ranged.add(SMGLLabel, c);
309 c.gridx = 2;
310 ranged.add(mgsmsgSpinner, c);
311 c.gridx = 1;
312 c.gridy = 4;
313 ranged.add(shotgunLabel, c);
314 c.gridx = 2;
315 ranged.add(shotgunSpinner, c);
316 c.gridx = 1;
317 c.gridy = 5;
318 ranged.add(assaultRifleLabel, c);
319 c.gridx = 2;
320 ranged.add(assaultRifleSpinner, c);
321 c.gridx = 1;
322 c.gridy = 6;
323 ranged.add(rangedEnerLabel, c);
324 c.gridx = 2;
325 ranged.add(rangedEnerSpinner, c);
326 c.gridx = 1;
327 c.gridy = 7;
328 ranged.add(flingShotLabel, c);
329 c.gridx = 2;
330 ranged.add(flingShotSpinner, c);
331 c.gridx = 1;
332 c.gridy = 8;
333 ranged.add(burstLabel, c);
334 c.gridx = 2;
335 ranged.add(burstSpinner, c);
336 c.gridx = 1;
337 c.gridy = 9;
338 ranged.add(fullAutoLabel, c);
339 c.gridx = 2;
340 ranged.add(fullAutoSpinner, c);
341 c.gridx = 1;
342 c.gridy = 10;
343 ranged.add(aimedShotLabel, c);
344 c.gridx = 2;
345 ranged.add(aimedShotSpinner, c);
346 c.gridx = 1;
347 c.gridy = 11;
348 ranged.add(multiRangedLabel, c);
349 c.gridx = 2;
350 ranged.add(multiRangedSpinner, c);
```

```

351     c.gridx = 1;
352     return ranged;
353 }
354 public JPanel createOtherWeapsPane()
355 {
356     otherWeaps = new JPanel(new GridBagLayout());
357     GridBagConstraints c = new GridBagConstraints();
358     JLabel sharpObjLabel = new JLabel("Sharp Objects:");
359     JLabel grenadeLabel = new JLabel("Grenade:");
360     JLabel heavyWeapsLabel = new JLabel("Heavy Weapons:");
361     SpinnerModel sharpObjModel = new SpinnerNumberModel(0, 0, 5000,
362 1);
363     SpinnerModel grenadeModel = new SpinnerNumberModel(0, 0, 5000,
364 1);
365     SpinnerModel heavyWeapsModel = new SpinnerNumberModel(0, 0, 500
0, 1);
366     JSpinner sharpObjSpinner = new JSpinner(sharpObjModel);
367     JSpinner grenadeSpinner = new JSpinner(grenadeModel);
368     JSpinner heavyWeapsSpinner = new JSpinner(heavyWeapsModel);
369     c.gridx = 1;
370     c.gridy = 0;
371     otherWeaps.add(sharpObjLabel, c);
372     c.gridx = 2;
373     otherWeaps.add(sharpObjSpinner, c);
374     c.gridx = 1;
375     c.gridy = 1;
376     otherWeaps.add(grenadeLabel, c);
377     c.gridx = 2;
378     otherWeaps.add(grenadeSpinner, c);
379     c.gridx = 1;
380     c.gridy = 2;
381     otherWeaps.add(heavyWeapsLabel, c);
382     c.gridx = 2;
383     otherWeaps.add(heavyWeapsSpinner, c);
384     c.gridx = 1;
385     return otherWeaps;
386 }
387 public JPanel createNanoPane()
388 {
389     nano = new JPanel(new GridBagLayout());
390     GridBagConstraints c = new GridBagConstraints();
391     JLabel matCreaLabel = new JLabel("Matter Creation:");
392     JLabel matMetLabel = new JLabel("Matter Matamorphosis:");
393     JLabel bioMetLabel = new JLabel("Biological Metamorphosis:");
394     JLabel timeSpaceLabel = new JLabel("Time and Space Alteration:");
395 );
396     JLabel senseImpLabel = new JLabel("Sensory Improvement:");
397     JLabel psyModLabel = new JLabel("Psychological Modifications:");
398 );
399     JLabel firstAidLabel = new JLabel("First Aid:");
400     SpinnerModel matCreaModel = new SpinnerNumberModel(0, 0, 5000,
401 1);
402     SpinnerModel matMetModel = new SpinnerNumberModel(0, 0, 5000, 1
);
403     SpinnerModel bioMetModel = new SpinnerNumberModel(0, 0, 5000, 1
);

```

```

399     SpinnerModel timeSpaceModel = new SpinnerNumberModel(0, 0, 5000
, 1);
400     SpinnerModel senseImpModel = new SpinnerNumberModel(0, 0, 5000,
1);
401     SpinnerModel psyModModel = new SpinnerNumberModel(0, 0, 5000, 1
);
402     SpinnerModel firstAidModel = new SpinnerNumberModel(0, 0, 5000,
1);
403     JSpinner matCreaSpinner = new JSpinner(matCreaModel);
404     JSpinner matMetSpinner = new JSpinner(matMetModel);
405     JSpinner bioMetSpinner = new JSpinner(bioMetModel);
406     JSpinner timeSpaceSpinner = new JSpinner(timeSpaceModel);
407     JSpinner senseImpSpinner = new JSpinner(senseImpModel);
408     JSpinner psyModSpinner = new JSpinner(psyModModel);
409     JSpinner firstAidSpinner = new JSpinner(firstAidModel);
410     c.gridx = 1;
411     c.gridy = 0;
412     nano.add(matCreaLabel, c);
413     c.gridx = 2;
414     nano.add(matCreaSpinner, c);
415     c.gridx = 1;
416     c.gridy = 1;
417     nano.add(matMetLabel, c);
418     c.gridx = 2;
419     nano.add(matMetSpinner, c);
420     c.gridx = 1;
421     c.gridy = 2;
422     nano.add(bioMetLabel, c);
423     c.gridx = 2;
424     nano.add(bioMetSpinner, c);
425     c.gridx = 1;
426     c.gridy = 3;
427     nano.add(timeSpaceLabel, c);
428     c.gridx = 2;
429     nano.add(timeSpaceSpinner, c);
430     c.gridx = 1;
431     c.gridy = 4;
432     nano.add(senseImpLabel, c);
433     c.gridx = 2;
434     nano.add(senseImpSpinner, c);
435     c.gridx = 1;
436     c.gridy = 5;
437     nano.add(psyModLabel, c);
438     c.gridx = 2;
439     nano.add(psyModSpinner, c);
440     c.gridx = 1;
441     c.gridy = 6;
442     nano.add(firstAidLabel, c);
443     c.gridx = 2;
444     nano.add(firstAidSpinner, c);
445     c.gridx = 1;
446     return nano;
447 }
448 public JPanel createSpeedPane()
449 {
450     speed = new JPanel(new GridBagLayout());

```

```

451 GridBagConstraints c = new GridBagConstraints();
452 JLabel evadeClsLabel = new JLabel("Evade Close-Combat:");
453 JLabel dodgeRngLabel = new JLabel("Dodge-Ranged:");
454 JLabel duckExpLabel = new JLabel("Duck Explosives:");
455 JLabel physInitLabel = new JLabel("Physical Initiative:");
456 JLabel meleeInitLabel = new JLabel("Melee Initiative:");
457 JLabel rangedInitLabel = new JLabel("Ranged Initiative:");
458 JLabel nanoInitLabel = new JLabel("Nano Initiative:");
459 JLabel nanoResLabel = new JLabel("Nano Resist:");
460 SpinnerModel evadeClsModel = new SpinnerNumberModel(0, 0, 5000,
1);
461 SpinnerModel dodgeRngModel = new SpinnerNumberModel(0, 0, 5000,
1);
462 SpinnerModel duckExpModel = new SpinnerNumberModel(0, 0, 5000,
1);
463 SpinnerModel physInitModel = new SpinnerNumberModel(0, 0, 5000,
1);
464 SpinnerModel meleeInitModel = new SpinnerNumberModel(0, 0, 5000
, 1);
465 SpinnerModel rangedInitModel = new SpinnerNumberModel(0, 0, 500
0, 1);
466 SpinnerModel nanoInitModel = new SpinnerNumberModel(0, 0, 5000,
1);
467 SpinnerModel nanoResModel = new SpinnerNumberModel(0, 0, 5000,
1);
468 JSpinner evadeClsSpinner = new JSpinner(evadeClsModel);
469 JSpinner dodgeRngSpinner = new JSpinner(dodgeRngModel);
470 JSpinner duckExpSpinner = new JSpinner(duckExpModel);
471 JSpinner physInitSpinner = new JSpinner(physInitModel);
472 JSpinner meleeInitSpinner = new JSpinner(meleeInitModel);
473 JSpinner rangedInitSpinner = new JSpinner(rangedInitModel);
474 JSpinner nanoInitSpinner = new JSpinner(nanoInitModel);
475 JSpinner nanoResSpinner = new JSpinner(nanoResModel);
476 c.gridx = 1;
477 c.gridy = 0;
478 speed.add(evadeClsLabel, c);
479 c.gridx = 2;
480 speed.add(evadeClsSpinner, c);
481 c.gridx = 1;
482 c.gridy = 1;
483 speed.add(dodgeRngLabel, c);
484 c.gridx = 2;
485 speed.add(dodgeRngSpinner, c);
486 c.gridx = 1;
487 c.gridy = 2;
488 speed.add(duckExpLabel, c);
489 c.gridx = 2;
490 speed.add(duckExpSpinner, c);
491 c.gridx = 1;
492 c.gridy = 3;
493 speed.add(physInitLabel, c);
494 c.gridx = 2;
495 speed.add(physInitSpinner, c);
496 c.gridx = 1;
497 c.gridy = 4;
498 speed.add(meleeInitLabel, c);

```

```

499     c.gridx = 2;
500     speed.add(meleeInitSpinner, c);
501     c.gridx = 1;
502     c.gridy = 5;
503     speed.add(rangedInitLabel, c);
504     c.gridx = 2;
505     speed.add(rangedInitSpinner, c);
506     c.gridx = 1;
507     c.gridy = 6;
508     speed.add(nanoInitLabel, c);
509     c.gridx = 2;
510     speed.add(nanoInitSpinner, c);
511     c.gridx = 1;
512     c.gridy = 7;
513     speed.add(nanoResLabel, c);
514     c.gridx = 2;
515     speed.add(nanoResSpinner, c);
516     c.gridx = 1;
517     return speed;
518 }
519 public JPanel createIPInfo()
520 {
521     IPInfo = new JPanel();
522     Integer IP = 200;
523     JLabel remainingIPLLabel = new JLabel("Remaining IP: " + IP);
524     //int spentIP =;
525     //int totalIP = IP + spentIP;
526     JLabel spentIPLLabel = new JLabel("Spent IP: ");
527     IPInfo.add(remainingIPLLabel);
528     return IPInfo;
529 }
530 public void actionPerformed(ActionEvent e)
531 {
532     if (e.getSource() == exit) {ipWindow.setVisible(false);}
533     if (e.getSource() == saveChar)
534     {
535
536     }
537 }
538 public void stateChanged(ChangeEvent e)
539 {
540     if (e.getSource() == strSpinnerModel)
541     {
542         str = (Integer) strSpinnerModel.getValue();
543         System.out.println(str);
544         //breakpoint
545     }
546 }
547 public static void createAndShowIPWindow()
548 {
549     JFrame.setDefaultLookAndFeelDecorated(true);
550     ipWindow = new JFrame("Allocate IP...");
551     ipWindow.setIconImage(new ImageIcon("Images/mainLogo.png").getI
mage());;
552     IPWindow test = new IPWindow();
553     JPanel contentPane = new JPanel();

```

```

554     ipWindow.setContentPane(contentPane);
555     ipWindow.setJMenuBar(test.createMenuBar());
556
557     //Populate the tabbed pane.
558     JTabbedPane tabbedPane = new JTabbedPane();
559
560     tabbedPane.addTab("Main Attributes", test.createMainAttPane());
561     tabbedPane.addTab("Body", test.createBodyPane());
562     tabbedPane.addTab("Melee", test.createMeleePane());
563     tabbedPane.addTab("Ranged", test.createRangedPane());
564     tabbedPane.addTab("Other Weapons", test.createOtherWeapsPane())
565     ;
566     tabbedPane.addTab("Nano and Aiding", test.createNanoPane());
567     tabbedPane.addTab("Evades and Initiatives", test.createSpeedPan
568     e());
569     tabbedPane.addTab("IP Info", test.createIPInfo());
570     tabbedPane.setTabPlacement(JTabbedPane.LEFT);
571     contentPane.add(tabbedPane);
572
573     ipWindow.pack();
574     ipWindow.setVisible(true);
575 }
576
577 public static void main(String[] args) {
578     try {
579         // Set System L&F
580         UIManager.setLookAndFeel(
581             UIManager.getSystemLookAndFeelClassName());
582     }
583     catch (UnsupportedLookAndFeelException e) {
584         // handle exception
585     }
586     catch (ClassNotFoundException e) {
587         // handle exception
588     }
589     catch (InstantiationException e) {
590         // handle exception
591     }
592     catch (IllegalAccessException e) {
593         // handle exception
594     }
595     //Schedule a job for the event-dispatching thread:
596     //creating and showing this application's GUI.
597     javax.swing.SwingUtilities.invokeLater(new Runnable() {
598         public void run() {
599             createAndShowIPWindow();
600         }
601     });
602 }

```

```

001 package playable;
002
003 import java.util.Random;
004
005 public class Specials {
006     int FAdmg, burstDmg, flingDmg, fastAtkDmg, ASDmg, brawlDmg, dimac
hDmg, snkAtkDmg, randomInt, multiplier;
007
008     public int FullAuto(int AAO, int FASkill, int clipSize, int minDm
g, int maxDmg, int MBS, int OppDef, int addDmg)
009     {
010         //Declare some variables
011         FAdmg = 0;
012         int capLanded = 0;
013         boolean cont = true;
014         //Determine the max number of bullets that can hit
015         if (FASkill < 500) {capLanded = 10;}
016         if (FASkill >=500 && FASkill < 1000) {capLanded = 15;}
017         if (FASkill >= 1000 && FASkill < 1250) {capLanded = 20;}
018         if (FASkill >= 1250) {capLanded = 25;}
019         int landed = 0;
020         Random generator = new Random();
021         //Find how many bullets landed
022         while (landed < clipSize && landed < capLanded && cont == true)
023         {
024             for(int landTest1 = 0; landTest1 < 10; landTest1++)
025             {
026                 if (AAO >= OppDef && landed <= 10) {landed++;}
027                 else {
028                     cont = false;
029                     break;
030                 }
031             }
032             for(int landTest2 = 0; landTest2 <=5 && landed < capLanded &&
cont == true; landTest2++)
033             {
034                 if (AAO*0.9 >= OppDef && landed <=15) {landed++;}
035             }
036             for(int landTest3 = 0; landTest3 <10 && landed < capLanded &&
cont == true; landTest3++)
037             {
038                 if (AAO*0.7 >= OppDef && landed < 25) {landed++;}
039             }
040         }
041         int damage = 0;
042         for(int i = 0; i < landed; i++)
043         {
044             damage = generator.nextInt(maxDmg-minDmg);
045             damage = damage + minDmg + addDmg;
046             if (AAO >= MBS) {damage = damage + minDmg;}
047             FAdmg = FAdmg + damage;
048         }
049         if (FAdmg > 15000) {FAdmg = 15000;}
050         return FAdmg;
051     }

```

```

052  public int Burst(int AAO, int burstSkill, int minDmg, int maxDmg,
    int MBS, int OppDef, int addDmg)
053  {
054      burstDmg = 0;
055      Random generator = new Random();
056      double randomMod = generator.nextDouble();
057      double mod = 1 + randomMod;
058      int landed = 0;
059      int counter = 0;
060      while (counter < 3)
061      {
062          if (AAO >= OppDef) {landed++;}
063          else if ((AAO*mod) >= OppDef) {landed++;}
064          counter++;
065      }
066      int damage = 0;
067      for(int i = 0; i < landed; i++)
068      {
069          damage = generator.nextInt(maxDmg-minDmg);
070          damage = damage + minDmg + addDmg;
071          if (AAO >= MBS) {damage = damage + minDmg;}
072          burstDmg = burstDmg + damage;
073      }
074      if (burstDmg > 13000) {burstDmg = 13000;}
075      return burstDmg;
076  }
077  public int FlingShot(int AAO, int flingSkill, int minDmg, int max
Dmg, int MBS, int oppDef, int addDmg)
078  {
079      double damage = 0;
080      Random generator = new Random();
081      damage = generator.nextInt(maxDmg-minDmg);
082      damage = damage + minDmg + addDmg;
083      if (AAO >= MBS) {damage = damage + minDmg;}
084      double flingDmgD = damage * 1.2;
085      return flingDmg = (int)flingDmgD;
086  }
087  public int AimedShot(int AAO, int ASSkill, int maxDmg, int MBS, i
nt addDmg)
088  {
089      Random generator = new Random();
090      int[] multipliers = new int[18];
091      multipliers[0] = 3;
092      multipliers[1] = 3;
093      multipliers[2] = 3;
094      multipliers[3] = 3;
095      multipliers[4] = 3;
096      multipliers[5] = 3;
097      multipliers[6] = 3;
098      multipliers[7] = 3;
099      multipliers[8] = 3;
100     multipliers[9] = 3;
101     multipliers[10] = 3;
102     multipliers[11] = 3;
103     multipliers[12] = 3;
104     multipliers[13] = 3;

```

```
105 multipliers[14] = 3;
106 multipliers[15] = 3;
107 multipliers[16] = 3;
108 multipliers[17] = 3;
109 if (ASSkill >=1700)
110 {
111     multipliers[0] = 3;
112     multipliers[1] = 4;
113     multipliers[2] = 5;
114     multipliers[3] = 6;
115     multipliers[4] = 7;
116     multipliers[5] = 8;
117     multipliers[6] = 9;
118     multipliers[7] = 10;
119     multipliers[8] = 11;
120     multipliers[9] = 12;
121     multipliers[10] = 13;
122     multipliers[11] = 14;
123     multipliers[12] = 15;
124     multipliers[13] = 16;
125     multipliers[14] = 17;
126     multipliers[15] = 18;
127     multipliers[16] = 19;
128     multipliers[17] = 20;
129 }
130 else if(ASSkill >=1600)
131 {
132     multipliers[0] = 3;
133     multipliers[1] = 4;
134     multipliers[2] = 5;
135     multipliers[3] = 6;
136     multipliers[4] = 7;
137     multipliers[5] = 8;
138     multipliers[6] = 9;
139     multipliers[7] = 10;
140     multipliers[8] = 11;
141     multipliers[9] = 12;
142     multipliers[10] = 13;
143     multipliers[11] = 14;
144     multipliers[12] = 15;
145     multipliers[13] = 16;
146     multipliers[14] = 17;
147     multipliers[15] = 18;
148     multipliers[16] = 19;
149 }
150 else if(ASSkill >=1500)
151 {
152     multipliers[0] = 3;
153     multipliers[1] = 4;
154     multipliers[2] = 5;
155     multipliers[3] = 6;
156     multipliers[4] = 7;
157     multipliers[5] = 8;
158     multipliers[6] = 9;
159     multipliers[7] = 10;
160     multipliers[8] = 11;
```

```
161     multipliers[9] = 12;
162     multipliers[10] = 13;
163     multipliers[11] = 14;
164     multipliers[12] = 15;
165     multipliers[13] = 16;
166     multipliers[14] = 17;
167     multipliers[15] = 18;
168 }
169 else if(ASSkill >=1400)
170 {
171     multipliers[0] = 3;
172     multipliers[1] = 4;
173     multipliers[2] = 5;
174     multipliers[3] = 6;
175     multipliers[4] = 7;
176     multipliers[5] = 8;
177     multipliers[6] = 9;
178     multipliers[7] = 10;
179     multipliers[8] = 11;
180     multipliers[9] = 12;
181     multipliers[10] = 13;
182     multipliers[11] = 14;
183     multipliers[12] = 15;
184     multipliers[13] = 16;
185     multipliers[14] = 17;
186 }
187 else if(ASSkill >=1300)
188 {
189     multipliers[0] = 3;
190     multipliers[1] = 4;
191     multipliers[2] = 5;
192     multipliers[3] = 6;
193     multipliers[4] = 7;
194     multipliers[5] = 8;
195     multipliers[6] = 9;
196     multipliers[7] = 10;
197     multipliers[8] = 11;
198     multipliers[9] = 12;
199     multipliers[10] = 13;
200     multipliers[11] = 14;
201     multipliers[12] = 15;
202     multipliers[13] = 16;
203 }
204 else if(ASSkill >=1200)
205 {
206     multipliers[0] = 3;
207     multipliers[1] = 4;
208     multipliers[2] = 5;
209     multipliers[3] = 6;
210     multipliers[4] = 7;
211     multipliers[5] = 8;
212     multipliers[6] = 9;
213     multipliers[7] = 10;
214     multipliers[8] = 11;
215     multipliers[9] = 12;
216     multipliers[10] = 13;
```

```
217     multipliers[11] = 14;
218     multipliers[12] = 15;
219 }
220 else if(ASSkill >=1100)
221 {
222     multipliers[0] = 3;
223     multipliers[1] = 4;
224     multipliers[2] = 5;
225     multipliers[3] = 6;
226     multipliers[4] = 7;
227     multipliers[5] = 8;
228     multipliers[6] = 9;
229     multipliers[7] = 10;
230     multipliers[8] = 11;
231     multipliers[9] = 12;
232     multipliers[10] = 13;
233     multipliers[11] = 14;
234 }
235 else if(ASSkill >=1000)
236 {
237     multipliers[0] = 3;
238     multipliers[1] = 4;
239     multipliers[2] = 5;
240     multipliers[3] = 6;
241     multipliers[4] = 7;
242     multipliers[5] = 8;
243     multipliers[6] = 9;
244     multipliers[7] = 10;
245     multipliers[8] = 11;
246     multipliers[9] = 12;
247     multipliers[10] = 13;
248 }
249 else if(ASSkill >=900)
250 {
251     multipliers[0] = 3;
252     multipliers[1] = 4;
253     multipliers[2] = 5;
254     multipliers[3] = 6;
255     multipliers[4] = 7;
256     multipliers[5] = 8;
257     multipliers[6] = 9;
258     multipliers[7] = 10;
259     multipliers[8] = 11;
260     multipliers[9] = 12;
261 }
262 else if(ASSkill >=800)
263 {
264     multipliers[0] = 3;
265     multipliers[1] = 4;
266     multipliers[2] = 5;
267     multipliers[3] = 6;
268     multipliers[4] = 7;
269     multipliers[5] = 8;
270     multipliers[6] = 9;
271     multipliers[7] = 10;
272     multipliers[8] = 11;
```

```
273     }
274     else if(ASSkill >=700)
275     {
276         multipliers[0] = 3;
277         multipliers[1] = 4;
278         multipliers[2] = 5;
279         multipliers[3] = 6;
280         multipliers[4] = 7;
281         multipliers[5] = 8;
282         multipliers[6] = 9;
283         multipliers[7] = 10;
284     }
285     else if(ASSkill >=600)
286     {
287         multipliers[0] = 3;
288         multipliers[1] = 4;
289         multipliers[2] = 5;
290         multipliers[3] = 6;
291         multipliers[4] = 7;
292         multipliers[5] = 8;
293         multipliers[6] = 9;
294     }
295     else if(ASSkill >=500)
296     {
297         multipliers[0] = 3;
298         multipliers[1] = 4;
299         multipliers[2] = 5;
300         multipliers[3] = 6;
301         multipliers[4] = 7;
302         multipliers[5] = 8;
303     }
304     else if(ASSkill >=400)
305     {
306         multipliers[0] = 3;
307         multipliers[1] = 4;
308         multipliers[2] = 5;
309         multipliers[3] = 6;
310         multipliers[4] = 7;
311     }
312     else if(ASSkill >=300)
313     {
314         multipliers[0] = 3;
315         multipliers[1] = 4;
316         multipliers[2] = 5;
317         multipliers[3] = 6;
318     }
319     else if(ASSkill >=200)
320     {
321         multipliers[0] = 3;
322         multipliers[1] = 4;
323         multipliers[2] = 5;
324     }
325     else if(ASSkill >=100)
326     {
327         multipliers[0] = 3;
328         multipliers[1] = 4;
```

```
329     }
330     else
331     {
332         multipliers[0] = 3;
333     }
334
335
336     int randomInt = generator.nextInt(100);
337     int multiplier = 3;
338     if(randomInt == 99)
339     {
340         multiplier = multipliers[17];
341     }
342     if(randomInt == 98)
343     {
344         multiplier = multipliers[16];
345     }
346     if(randomInt == 97)
347     {
348         multiplier = multipliers[15];
349     }
350     if(randomInt == 96)
351     {
352         multiplier = multipliers[14];
353     }
354     if(randomInt == 95 || randomInt == 94)
355     {
356         multiplier = multipliers[13];
357     }
358     if(randomInt == 93 || randomInt == 92)
359     {
360         multiplier = multipliers[12];
361     }
362     if(randomInt < 92 && randomInt >= 89)
363     {
364         multiplier = multipliers[11];
365     }
366     if(randomInt < 89 && randomInt >= 86)
367     {
368         multiplier = multipliers[10];
369     }
370     if(randomInt < 86 && randomInt >= 83)
371     {
372         multiplier = multipliers[9];
373     }
374     if(randomInt < 83 && randomInt >= 79)
375     {
376         multiplier = multipliers[8];
377     }
378     if(randomInt < 79 && randomInt >= 75)
379     {
380         multiplier = multipliers[7];
381     }
382     if(randomInt < 75 && randomInt >= 70)
383     {
384         multiplier = multipliers[6];
```

```

385     }
386     if(randomInt < 70 && randomInt >= 64)
387     {
388         multiplier = multipliers[5];
389     }
390     if(randomInt < 64 && randomInt >= 57)
391     {
392         multiplier = multipliers[4];
393     }
394     if(randomInt < 57 && randomInt >= 49)
395     {
396         multiplier = multipliers[3];
397     }
398     if(randomInt < 49 && randomInt >= 39)
399     {
400         multiplier = multipliers[2];
401     }
402     if(randomInt < 39 && randomInt >= 24)
403     {
404         multiplier = multipliers[1];
405     }
406     if(randomInt < 24 && randomInt >= 0)
407     {
408         multiplier = multipliers[0];
409     }
410     else
411     {
412         multiplier = 3;
413     }
414
415
416
417     int ASDmg = (maxDmg * 3) * multiplier;
418     return ASDmg;
419 }
420 public int Brawl(int AAO, int minDmg, int maxDmg, int oppDef, int
addDmg)
421 {
422     if (AAO >= oppDef)
423     {
424         double brawlDmgdbl = maxDmg * 1.1;
425         brawlDmg = (int) brawlDmgdbl;
426     }
427     else {brawlDmg = 0;}
428     return brawlDmg;
429 }
430 public int FastAttack(int AAO, int fastAtkSkill, int minDmg, int
maxDmg, int MBS, int oppDef, int addDmg)
431 {
432     double damage = 0;
433     Random generator = new Random();
434     damage = generator.nextInt(maxDmg-minDmg);
435     damage = damage + minDmg + addDmg;
436     if (AAO >= MBS) {damage = damage + minDmg;}
437     double fastAtkDmgD = damage * 1.2;
438     return fastAtkDmg = (int)fastAtkDmgD;

```

```
439     }
440     public int Dimach(int AAO, int minDmg, int maxDmg, int oppDef, int
addDmg)
441     {
442         if ((AAO * 1.5) > oppDef)
443         {
444             Random generator = new Random();
445             int damage = generator.nextInt(maxDmg - minDmg);
446             damage = damage + minDmg + addDmg;
447             dimachDmg = damage * 10;
448         }
449         else {dimachDmg = 0;}
450         return dimachDmg;
451     }
452     public int SneakAttack(int maxDmg)
453     {
454         snkAtkDmg= maxDmg * 6;
455         return snkAtkDmg;
456     }
457 }
```

```

/*=====
====
Class:      Main                      Program:  AOText
Author:    Jory Stewart
Date:     March 30, 2008                Teacher:  Gerry
Donaldson
School:    Sir Winston Churchill High School, Calgary, Alberta, Canada
Language:  Java J2SE 5.0              Target Operating System: Java Virtual
Machine
System:    Pentium IV 2.5 GHz running under Windows XP      IDE:
Eclipse 3.1
=====
====*/

001 package playable;
002
003 import java.awt.*;
004 import java.io.*;
005 import java.awt.event.*;
006
007 import javax.swing.*;
008 import javax.swing.filechooser.*;
009 import javax.swing.filechooser.FileFilter;
010
011 /** @author Jory Stewart
012  * This is the class you run to start the program.
013  */
014 public class Main implements ActionListener{
015     JMenuItem close, newChar, loadChar, saveChar, openIPWindow, loadO
pp, oppProf, aboutHelp, differences, versionNotes, about, contact;
016     JPanel statsPanel, weaponsPanel, rangedSpecialsPanel, meleeSpecia
lsPanel, specialTimers;
017     JLabel hpLabel, nanoLabel, levelLabel, xpLabel, currentWeapon, ch
ngWeapon, fullAutoCycleLabel, burstCycleLabel, flingShotCycleLabel, aim
edShotCycleLabel, fastAttackCycleLabel, brawlCycleLabel, dimachCycleLab
el, sneakAttackCycleLabel, backstabCycleLabel;
018     JButton fullAuto, burst, flingShot, aimedShot, fastAttack, brawl,
dimach, sneakAttack, startAtk, hb1, hb2, hb3, hb4, hb5, hb6, hb7, hb8,
hb9, hb0;
019     JTextArea combatPane;
020     playable.Specials specials = new Specials();
021     static JFrame frame;
022     final JFileChooser fc = new JFileChooser();
023
024     /**
025      * This method creates the JMenuBar used for the main frame.
026      * @return The complete JMenuBar
027      */
028     public JMenuBar createMenuBar()
029     {
030         //Creates the menu bar.
031         JMenuBar menubar = new JMenuBar();
032         //Create the first menu in the bar, "File"
033         JMenu file = new JMenu("File");
034         //Create the second menu, "Character"

```

```

035 JMenu character = new JMenu("Character");
036 //Create the third menu, "Opponent"
037 JMenu opponent = new JMenu("Opponent");
038 //Create the fourth menu, "Help"
039 JMenu help = new JMenu("Help");
040 //Add all the menus to the menu bar.
041 menubar.add(file);
042 menubar.add(character);
043 menubar.add(opponent);
044 menubar.add(help);
045
046 //Items in the first menu, "File"
047 close = new JMenuItem("Exit");
048 close.setMnemonic(KeyEvent.VK_F4);
049 close.addActionListener(this);
050 //Populate the first menu, "File"
051 file.add(close);
052
053 //Items in the second menu, "Character"
054 newChar = new JMenuItem("Create New Character");
055 newChar.addActionListener(this);
056 loadChar = new JMenuItem("Load existing character...");
057 loadChar.addActionListener(this);
058 saveChar = new JMenuItem("Save");
059 saveChar.addActionListener(this);
060 openIPWindow = new JMenuItem("Allocate IP...");
061 openIPWindow.addActionListener(this);
062 //Populate the second menu, "Character"
063 character.add(newChar);
064 character.addSeparator();
065 character.add(loadChar);
066 character.add(saveChar);
067 character.addSeparator();
068 character.add(openIPWindow);
069
070 //Items in the third menu, "Opponent"
071 loadOpp = new JMenuItem("Load character file for opponent...");
072 loadOpp.addActionListener(this);
073 oppProf = new JMenuItem("Configure Opponent...");
074 oppProf.addActionListener(this);
075 //Populate the third menu, "Opponent"
076 opponent.add(loadOpp);
077 opponent.add(oppProf);
078
079 //Items in the fourth menu, "Help"
080 aboutHelp = new JMenuItem("Help");
081 aboutHelp.addActionListener(this);
082 differences = new JMenuItem("AO/AOText Differences");
083 differences.addActionListener(this);
084 versionNotes = new JMenuItem("Version Notes");
085 versionNotes.addActionListener(this);
086 about = new JMenuItem("About AOText");
087 about.addActionListener(this);
088 contact = new JMenuItem("Contact Me");
089 contact.addActionListener(this);
090 //Populate the fourth menu, "Help"

```



```

plemented)
143     chngWeapon = new JLabel("Select Weapon: ");
144     String[] weaponsList = {"Ofab Mongoose Mk I [1HE]", "Ofab Wolf
Mk I [2HE]", "Ofab Panther Mk I [1HB]", "Ofab Bear Mk I [2HB]", "Ofab V
iper Mk I [Piercing]", "Dreadloch Shen Sticks [MA]", "Dreadlock Reaper
[Melee Energy]", "Ofab Peregrine Mk I [Pistol]", "Ofab Shark Mk I [Assa
ult Rifle]", "Ofab Silverback Mk I [Shotgun]", "Ofab Cobra Mk I [Rifle]
", "Ofab Hawk Mk I [SMG]", "Ofab Tiger Mk I [Bow]", "Dreadloch Lancer [
Ranged Energy]", "Dreadloch Cannon [Heavy Weapons]", "Dreadloch Shruike
n [Sharp Objects]", "Dreadloch Redliner [Grenade]", "Basic Cyberdeck [Na
no-Technician only, no attack skill]"};
145     JComboBox weaponSelection = new JComboBox(weaponsList);
146     //Populate the Weapons... Panel
147     a.gridy = 0;
148     weaponsPanel.add(currentWeapon, a);
149     //weaponsPanel.add(currentWeapLabel);
150     a.gridy = 2;
151     weaponsPanel.add(chngWeapon, a);
152     a.gridx = 1;
153     weaponsPanel.add(weaponSelection, a);
154     //Adds a border around the Weapons Panel
155     weaponsPanel.setBorder(BorderFactory.createCompoundBorder(
156     BorderFactory.createTitledBorder("Weapons"),
157     BorderFactory.createEmptyBorder(5,5,5,5)));
158     return weaponsPanel;
159 }
160 /**
161  * A panel with buttons for activating the ranged special attacks
162  * @return The panel containing buttons for Full Auto, Burst, Fli
ng Shot, and Aimed Shot.
163  */
164 public JPanel createRangedSpecialsPanel()
165 {
166     ImageIcon fullAutoIcon = new ImageIcon("Images/fullAuto.jpg");
167     ImageIcon burstIcon = new ImageIcon("Images/burst.jpg");
168     ImageIcon flingShotIcon = new ImageIcon("Images/flingShot.jpg")
;
169     ImageIcon aimedShotIcon = new ImageIcon("Images/aimedShot.jpg")
;
170
171     //Creates the RangedSpecialsPanel
172     rangedSpecialsPanel = new JPanel();
173     //Creates the Ranged Special toolbar.
174     JToolBar rangedSpecials = new JToolBar();
175     //Creates the buttons for the Ranged Special toolbar.
176     fullAuto = new JButton(fullAutoIcon);
177     fullAuto.setToolTipText("Full Auto");
178     fullAuto.setMnemonic(KeyEvent.VK_COMMA);
179     fullAuto.addActionListener(this);
180     burst = new JButton(burstIcon);
181     burst.setToolTipText("Burst");
182     burst.setMnemonic(KeyEvent.VK_M);
183     burst.addActionListener(this);
184     flingShot = new JButton(flingShotIcon);
185     flingShot.setToolTipText("Fling Shot");

```

```

186     flingShot.setMnemonic(KeyEvent.VK_L);
187     flingShot.addActionListener(this);
188     aimedShot = new JButton(aimedShotIcon);
189     aimedShot.setToolTipText("Aimed Shot");
190     aimedShot.setMnemonic(KeyEvent.VK_O);
191     aimedShot.addActionListener(this);
192     //Add the buttons to the Ranged Specials toolbar.
193     rangedSpecials.add(fullAuto);
194     rangedSpecials.add(burst);
195     rangedSpecials.add(flingShot);
196     rangedSpecials.add(aimedShot);
197     rangedSpecials.setFloatable(false);
198 //     Adds a border around the RangedSpecialsPanel
199     rangedSpecialsPanel.setBorder(BorderFactory.createCompoundBorder(
200         BorderFactory.createTitledBorder("Ranged Specials"),
201         BorderFactory.createEmptyBorder(5,5,5,5)));
202     //Adds the Ranged Specials toolbar to the RangedSpecialsPanel.
203     rangedSpecialsPanel.add(rangedSpecials);
204     return rangedSpecialsPanel;
205 }
206 /**
207  * A panel with buttons for activating the melee special attacks.
208  * @return The panel containing buttons for Fast Attack, Brawl, Dimach, and Sneak Attack.
209  */
210 public JPanel createMeleeSpecialsPanel()
211 {
212     ImageIcon brawlIcon = new ImageIcon("Images/brawl.jpg");
213     ImageIcon fastAttackIcon = new ImageIcon("Images/fastAttack.jpg");
214     ImageIcon dimachIcon = new ImageIcon("Images/dimach.jpg");
215     ImageIcon sneakAttackIcon = new ImageIcon("Images/sneakAttack.jpg");
216 //     Creates the MeleeSpecialsPanel
217     meleeSpecialsPanel = new JPanel();
218     //Creates the Melee Specials toolbar.
219     JToolBar meleeSpecials = new JToolBar();
220     //Creates the buttons for the Melee Special toolbar.
221     fastAttack = new JButton(fastAttackIcon);
222     fastAttack.setToolTipText("Fast Attack");
223     fastAttack.setMnemonic(KeyEvent.VK_N);
224     fastAttack.addActionListener(this);
225     brawl = new JButton(brawlIcon);
226     brawl.setToolTipText("Brawl");
227     brawl.setMnemonic(KeyEvent.VK_B);
228     brawl.addActionListener(this);
229     dimach = new JButton(dimachIcon);
230     dimach.setToolTipText("Dimach");
231     dimach.setMnemonic(KeyEvent.VK_K);
232     dimach.addActionListener(this);
233     sneakAttack = new JButton(sneakAttackIcon);
234     sneakAttack.setToolTipText("Sneak Attack");
235     sneakAttack.setMnemonic(KeyEvent.VK_J);
236     sneakAttack.addActionListener(this);
237     //Adds the buttons to the Melee Specials toolbar.

```

```

238     meleeSpecials.add(fastAttack);
239     meleeSpecials.add(brawl);
240     meleeSpecials.add(dimach);
241     meleeSpecials.add(sneakAttack);
242     meleeSpecials.setFloatable(false);
243 //     Adds a border around the MeleeSpecialsPanel
244     meleeSpecialsPanel.setBorder(BorderFactory.createCompoundBorder
(
245         BorderFactory.createTitledBorder("Melee Specials"),
246         BorderFactory.createEmptyBorder(5,5,5,5));
247     //Adds the Melee Specials toolbar to the MeleeSpecialsPanel.
248     meleeSpecialsPanel.add(meleeSpecials);
249     return meleeSpecialsPanel;
250 }
251 /**
252  * Creates a text area for the displaying combat messages.
253  * @return The editor pane for displaying combat messages.
254  */
255 public JScrollPane createCombatPane()
256 {
257 //     Create an editor pane.
258     combatPane = new JTextArea("Welcome to AOText.\n");
259     combatPane.setEditable(false);
260     JScrollPane editorScrollPane = new JScrollPane(combatPane);
261     editorScrollPane.setVerticalScrollBarPolicy(
262         JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
263     editorScrollPane.setPreferredSize(new Dimension(500,300));
264     editorScrollPane.setMinimumSize(new Dimension(500,300));
265     editorScrollPane.setBorder(BorderFactory.createCompoundBord
er(
266         BorderFactory.createTitledBorder("Combat Window"),
267         BorderFactory.createEmptyBorder(5,5,5,5));
268     return editorScrollPane;
269 }
270 /**
271  * Shows the time remaining until special attacks are usable agai
n.
272  * @return The panel displaying the remaining cooldown on special
attacks.
273  */
274 public JPanel createSpecialTimersPanel()
275 {
276 //     Creates the SpecialTimers panel.
277     specialTimers = new JPanel(new GridBagLayout());
278     GridBagConstraints f = new GridBagConstraints();
279     //Create the countdown labels.
280     Timer FACycle = new Timer(11000, this);
281     FACycle.setRepeats(false);
282     FACycle.start();
283     Timer burstCycle = new Timer(11000, this);
284     FACycle.setRepeats(false);
285     FACycle.start();
286     Timer flingShotCycle = new Timer(11000, this);
287     FACycle.setRepeats(false);
288     FACycle.start();
289     Timer aimedShotCycle = new Timer(11000, this);

```

```

290     FACycle.setRepeats(false);
291     FACycle.start();
292     Timer fastAttackCycle = new Timer(11000, this);
293     FACycle.setRepeats(false);
294     FACycle.start();
295     Timer brawlCycle = new Timer(11000, this);
296     FACycle.setRepeats(false);
297     FACycle.start();
298     Timer dimachCycle = new Timer(11000, this);
299     FACycle.setRepeats(false);
300     FACycle.start();
301     Timer sneakAttackCycle = new Timer(11000, this);
302     FACycle.setRepeats(false);
303     FACycle.start();
304     Timer backstabCycle = new Timer(11000, this);
305     FACycle.setRepeats(false);
306     FACycle.start();
307     fullAutoCycleLabel = new JLabel("Full Auto Available in: " + FA
Cycle.getDelay() + " sec");
308     burstCycleLabel = new JLabel("Burst Available in: " + "sec");
309     flingShotCycleLabel = new JLabel("Fling Shot Available in: " +
"sec");
310     aimedShotCycleLabel = new JLabel("Aimed Shot Available in: " +
"sec");
311     fastAttackCycleLabel = new JLabel("Fast Attack Available in: "
+ "sec");
312     brawlCycleLabel = new JLabel("Brawl Available in: " + "sec");
313     dimachCycleLabel = new JLabel("Dimach Available in: " + "sec");
314     sneakAttackCycleLabel = new JLabel("Sneak Attack Available in:
" + "sec");
315     backstabCycleLabel = new JLabel("Backstab Available in: " + "se
c");
316     //Populate the SpecialTimers panel
317     f.gridx = 0;
318     f.gridy = 0;
319     f.anchor = GridBagConstraints.LINE_START;
320     specialTimers.add(fullAutoCycleLabel, f);
321     f.gridy = 1;
322     specialTimers.add(burstCycleLabel, f);
323     f.gridy = 2;
324     specialTimers.add(flingShotCycleLabel, f);
325     f.gridy = 3;
326     specialTimers.add(aimedShotCycleLabel, f);
327     f.gridy = 4;
328     specialTimers.add(fastAttackCycleLabel, f);
329     f.gridy = 5;
330     specialTimers.add(brawlCycleLabel, f);
331     f.gridy = 6;
332     specialTimers.add(dimachCycleLabel, f);
333     f.gridy = 7;
334     specialTimers.add(sneakAttackCycleLabel, f);
335     f.gridy = 8;
336     specialTimers.add(backstabCycleLabel, f);
337     //Adds a border around the SpecialTimersPanel
338     specialTimers.setBorder(BorderFactory.createCompoundBorder(
BorderFactory.createTitledBorder("Special Timers"),
339

```

```

340     BorderFactory.createEmptyBorder(5,5,5,5));
341     return specialTimers;
342 }
343 /**
344  * Creates a hotbar of buttons that can be customized. This feature is currently disabled.
345  * @return A row of customizable buttons.
346  */
347 public JToolBar createHotBar()
348 {
349     // Creates the "Hotbar"
350     JToolBar hotbar = new JToolBar();
351     startAtk = new JButton("Commence Attack");
352     hb1 = new JButton("1");
353     hb2 = new JButton("2");
354     hb3 = new JButton("3");
355     hb4 = new JButton("4");
356     hb5 = new JButton("5");
357     hb6 = new JButton("6");
358     hb7 = new JButton("7");
359     hb8 = new JButton("8");
360     hb9 = new JButton("9");
361     hb0 = new JButton("0");
362     hotbar.add(startAtk);
363     hotbar.add(hb1);
364     hotbar.add(hb2);
365     hotbar.add(hb3);
366     hotbar.add(hb4);
367     hotbar.add(hb5);
368     hotbar.add(hb6);
369     hotbar.add(hb7);
370     hotbar.add(hb8);
371     hotbar.add(hb9);
372     hotbar.add(hb0);
373     hotbar.setFloatable(false);
374     return hotbar;
375 }
376 /**
377  * Outlines the actions taken when various items are interacted with.
378  */
379 public void actionPerformed(ActionEvent e) {
380
381     if (e.getSource() == close) {frame.setVisible(false);}
382     else if (e.getSource() == newChar) {
383         javax.swing.SwingUtilities.invokeLater(new Runnable(){
384             public void run(){
385                 playable.CreateChar.createAndShowCharCreation();});}
386     else if (e.getSource() == openIPWindow) {javax.swing.SwingUtilities.invokeLater(new Runnable(){
387         public void run(){
388             playable.IPWindow.createAndShowIPWindow();});}
389     else if (e.getSource() == differences) {javax.swing.SwingUtilities.invokeLater(new Runnable(){
390         public void run(){
391             playable.About.createAndShowDifferences();});}

```

```

392     else if (e.getSource() == versionNotes) {javax.swing.SwingUtili
ties.invokeLater(new Runnable(){
393         public void run(){
394             playable.About.createAndShowVersionNotes();}});}
395     else if (e.getSource() == about) {
396         javax.swing.SwingUtilities.invokeLater(new Runnable(){
397             public void run(){
398                 playable.About.createAndShowAbout();}});}
399     else if(e.getSource() == oppProf){
400         javax.swing.SwingUtilities.invokeLater(new Runnable(){
401             public void run(){
402                 playable.CreateChar.createAndShowOppCreation();}});}
403     else if(e.getSource() == contact){
404         javax.swing.SwingUtilities.invokeLater(new Runnable(){
405             public void run(){
406                 playable.About.showContactInfo();}});}
407 //else if(e.getSource() == saveChar) {
408 //javax.swing.SwingUtilities.invokeLater(new Runnable(){
409 //public void run(){
410 //playable.FileManipulations.
411 // }
412 // })
413 //}
414     else if(e.getSource() == loadChar || e.getSource() == loadOpp)
415     {
416         int returnVal = fc.showOpenDialog(frame);
417         if (returnVal == JFileChooser.APPROVE_OPTION)
418         {
419             File file = fc.getSelectedFile();
420         }
421     }
422     }
423     else if(e.getSource() == aboutHelp) {
424         javax.swing.SwingUtilities.invokeLater(new Runnable(){
425             public void run(){
426                 playable.About.createAndShowHelp();}});}
427     else if(e.getSource() == fullAuto)
428     {
429         specials.FullAuto(2000,1300,50,125,450,1000,500,0);
430         combatPane.append("You hit XXX for " + specials.FAdmg + " poin
ts of Full Auto damage.\n");
431     }
432     else if(e.getSource() == burst)
433     {
434         specials.Burst(2000,1300,125,450,1000,1900,0);
435         combatPane.append("You hit XXX for " + specials.burstDmg + "
points of Burst damage.\n");
436     }
437     else if(e.getSource() == flingShot)
438     {
439         specials.FlingShot(2000, 1300, 125, 450, 1000, 500, 0);
440         combatPane.append("You hit XXX for " + specials.flingDmg + "
points of Fling Shot damage.\n");
441     }
442     else if(e.getSource() == aimedShot)
443     {

```

```

444     specials.AimedShot(2000, 2000, 450, 1000, 0);
445     combatPane.append("You hit XXX for " + specials.ASDmg + " poi
nts of Aimed Shot damage.\n");
446     }
447     else if(e.getSource() == fastAttack)
448     {
449         specials.FastAttack(2000, 1300, 125, 450, 1000, 500, 0);
450         combatPane.append("You hit XXX for " + specials.fastAtkDmg +
" points of Fast Attack damage.\n");
451     }
452     else if(e.getSource() == brawl)
453     {
454         specials.Brawl(2000, 125, 450, 1900, 0);
455         combatPane.append("You hit XXX for " + specials.brawlDmg + "
points of Brawl damage.\n");
456     }
457     else if(e.getSource() == dimach)
458     {
459         specials.Dimach(2000, 125, 450, 1900, 0);
460         combatPane.append("You hit XXX for " + specials.dimachDmg + "
points of Dimach damage.\n");
461     }
462     else if(e.getSource() == sneakAttack)
463     {
464         specials.SneakAttack(450);
465         combatPane.append("You hit XXX for " + specials.snkAtkDmg + "
points of Sneak Attack damage.\n");
466     }
467     }
468     /**
469     * This method uses several previous methods to assemble the GUI.
470     */
471     public static void createAndShowGUI()
472     {
473         //Makes decorations
474         JFrame.setDefaultLookAndFeelDecorated(true);
475         //Creates the JFrame (top-level container)
476         frame = new JFrame("AOText");
477         //Sets the image used in the title
478         frame.setIconImage(new ImageIcon("Images/mainLogo.png").getImag
e());
479         //Makes the program exit when the Close button is clicked.
480         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
481         //Create a content pane.
482         JPanel contentPane = new JPanel(new GridBagLayout());
483         GridBagConstraints c = new GridBagConstraints();
484         frame.getContentPane().add(contentPane);
485         //Adds the menu bar to the frame.
486         Main test = new Main();
487         frame.setJMenuBar(test.createMenuBar());
488         //Populates the content pane.
489         c.gridx = 0;
490         c.gridy = 0;
491         c.insets = new Insets(1,1,5,5);
492         contentPane.add(test.createHPPanel(), c);
493         c.gridx = 1;

```

```

494     c.gridy = 0;
495     c.insets = new Insets(1,5,5,5);
496     contentPane.add(test.createWeaponsPanel(), c);
497     c.gridx = 2;
498     c.gridy = 0;
499     c.insets = new Insets(1,5,1,1);
500     contentPane.add(test.createRangedSpecialsPanel(), c);
501     c.gridx = 2;
502     c.gridy = 1;
503     contentPane.add(test.createMeleeSpecialsPanel(), c);
504         c.gridx = 0;
505         c.gridy = 1;
506         c.gridwidth = 2;
507         c.gridheight = 2;
508         c.weightx = 1.0;
509         c.weighty = 1.0;
510         c.insets = new Insets(5,5,5,5);
511         contentPane.add(test.createCombatPane(), c);
512         c.gridx = 2;
513     c.gridy = 2;
514     c.insets = new Insets(5,5,1,1);
515     contentPane.add(test.createSpecialTimersPanel(), c);
516     c.gridx = 0;
517     c.gridy = 3;
518     c.insets = new Insets(1,5,1,1);
519     contentPane.add(test.createHotBar(), c);
520     //Display the main frame.
521     frame.setBounds(25,25,750,750);
522     frame.setLocationRelativeTo(null);
523     frame.pack();
524     frame.setVisible(true);
525 }
526 /**
527  * Displays the GUI in a manner than maximizes thread safety.
528  * @param args
529  */
530 public static void main(String[] args) {
531     try {
532         // Set System L&F
533         UIManager.setLookAndFeel(
534             UIManager.getSystemLookAndFeelClassName());
535     }
536     catch (UnsupportedLookAndFeelException e) {
537         // handle exception
538     }
539     catch (ClassNotFoundException e) {
540         // handle exception
541     }
542     catch (InstantiationException e) {
543         // handle exception
544     }
545     catch (IllegalAccessException e) {
546         // handle exception
547     }
548     //Schedule a job for the event-dispatching thread:
549     //creating and showing this application's GUI.

```

```
550     javax.swing.SwingUtilities.invokeLater(new Runnable(){
551         public void run(){
552             createAndShowGUI();}});
553     javax.swing.SwingUtilities.invokeLater(new Runnable(){
554         public void run(){
555             playable.About.createAndShowAbout();}});
556
557
558     }
559 }
```