```
/* ===========================================================
 * JFreeChart : a free chart library for the Java(tm) platform
 * ===========================================================
 *
 * (C) Copyright 2000-2005, by Object Refinery Limited and Contributors.
 *
 * Project Info:  http://www.jfree.org/jfreechart/index.html
 *
 * This library is free software; you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
 * or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public
 * License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301,
 * USA.
 *
 * [Java is a trademark or registered trademark of Sun Microsystems, Inc.
 * in the United States and other countries.]
 *
 * ---------
 * Task.java
 * ---------
 * (C) Copyright 2003, 2004, by Object Refinery Limited.
 *
 * Original Author:  David Gilbert (for Object Refinery Limited);
 * Contributor(s):   -;
 *
 *
 * Changes
 * -------
 * 10-Jan-2003 : Version 1 (DG);
 * 16-Sep-2003 : Added percentage complete (DG);
 * 30-Jul-2004 : Added clone() and equals() methods and implemented
 *               Serializable (DG);
 *
 */

package org.jfree.data.gantt;

import java.io.Serializable;
import java.util.Date;
import java.util.List;

import org.jfree.data.time.SimpleTimePeriod;
import org.jfree.data.time.TimePeriod;
import org.jfree.util.ObjectUtilities;
import org.jfree.util.PublicCloneable;

/**
 * A simple representation of a task.  The task has a description and a
 * duration.  You can add sub-tasks to the task.
 */
public class Task implements Cloneable, PublicCloneable, Serializable {

    /** For serialization. */
    private static final long serialVersionUID = 1094303785346988894L;

    /** The task description. */
    private String description;

    /** The time period for the task (estimated or actual). */
```

C:/jfreechart-1.0.2/jfreechart-1.0.2/source/org/jfree/data/gantt/Task.java, Top line: 70

```
070     private TimePeriod duration;
071
072     /** The percent complete (<code>null</code> is permitted). */
073     private Double percentComplete;
074
075     /** Storage for the sub-tasks (if any). */
076     private List subtasks;
077
078     /**
079      * Creates a new task.
080      *
081      * @param description  the task description (<code>null</code> not
082      *                     permitted).
083      * @param duration  the task duration (<code>null</code> permitted).
084      */
085     public Task(String description, TimePeriod duration) {
086         if (description == null) {
087             throw new IllegalArgumentException("Null 'description' argument.");
088         }
089         this.description = description;
090         this.duration = duration;
091         this.percentComplete = null;
092         this.subtasks = new java.util.ArrayList();
093     }
094
095     /**
096      * Creates a new task.
097      *
098      * @param description  the task description (<code>null</code> not
099      *                     permitted).
100      * @param start  the start date (<code>null</code> not permitted).
101      * @param end  the end date (<code>null</code> not permitted).
102      */
103     public Task(String description, Date start, Date end) {
104         this(description, new SimpleTimePeriod(start, end));
105     }
106
107     /**
108      * Returns the task description.
109      *
110      * @return The task description (never <code>null</code>).
111      */
112     public String getDescription() {
113         return this.description;
114     }
115
116     /**
117      * Sets the task description.
118      *
119      * @param description  the description (<code>null</code> not permitted).
120      */
121     public void setDescription(String description) {
122         if (description == null) {
123             throw new IllegalArgumentException("Null 'description' argument.");
124         }
125         this.description = description;
126     }
127
128     /**
129      * Returns the duration (actual or estimated) of the task.
130      *
```

C:/jfreechart-1.0.2/jfreechart-1.0.2-modified/source/org/jfree/data/gantt/Task.java, Top line: 70

```
070     private TimePeriod duration;
071
072     /** The percent complete (<code>null</code> is permitted). */
073     private Double percentComplete;
074
075     /** Storage for the sub-tasks (if any). */
076     private List subtasks;
077
078     /** If it is a summary task. */
079     private boolean summaryTask;
080
081     /** If it is a summary task. */
082     private boolean milestone;
083
084     /**
085      * Creates a new task.
086      *
087      * @param description  the task description (<code>null</code> not
088      *                     permitted).
089      * @param duration  the task duration (<code>null</code> permitted).
090      */
091     public Task(String description, TimePeriod duration) {
092         if (description == null) {
093             throw new IllegalArgumentException("Null 'description' argument.");
094         }
095         this.description = description;
096         this.duration = duration;
097         this.percentComplete = null;
098         this.subtasks = new java.util.ArrayList();
099         this.summaryTask = false;
100         this.milestone = false;
101     }
102
103     /**
104      * Creates a new task.
105      *
106      * @param description  the task description (<code>null</code> not
107      *                     permitted).
108      * @param start  the start date (<code>null</code> not permitted).
109      * @param end  the end date (<code>null</code> not permitted).
110      */
111     public Task(String description, Date start, Date end) {
112         this(description, new SimpleTimePeriod(start, end));
113     }
114
115     /**
116      * Returns the task description.
117      *
118      * @return The task description (never <code>null</code>).
119      */
120     public String getDescription() {
121         return this.description;
122     }
123
124     /**
125      * Sets the task description.
126      *
127      * @param description  the description (<code>null</code> not permitted).
128      */
129     public void setDescription(String description) {
130         if (description == null) {
131             throw new IllegalArgumentException("Null 'description' argument.");
132         }
133         this.description = description;
134     }
135
136     /**
137      * Returns the duration (actual or estimated) of the task.
138      *
```

C:/jfreechart-1.0.2/jfreechart-1.0.2/source/org/jfree/data/gantt/Task.java, Top line: 131

```
131         * @return The task duration (possibly <code>null</code>).
132         */
133        public TimePeriod getDuration() {
134            return this.duration;
135        }
136
137        /**
138         * Sets the task duration (actual or estimated).
139         *
140         * @param duration  the duration (<code>null</code> permitted).
141         */
142        public void setDuration(TimePeriod duration) {
143            this.duration = duration;
144        }
145
146        /**
147         * Returns the percentage complete for this task.
148         *
149         * @return The percentage complete (possibly <code>null</code>).
150         */
151        public Double getPercentComplete() {
152            return this.percentComplete;
153        }
154
155        /**
156         * Sets the percentage complete for the task.
157         *
158         * @param percent  the percentage (<code>null</code> permitted).
159         */
160        public void setPercentComplete(Double percent) {
161            this.percentComplete = percent;
162        }
163
164        /**
165         * Sets the percentage complete for the task.
166         *
167         * @param percent  the percentage.
168         */
169        public void setPercentComplete(double percent) {
170            setPercentComplete(new Double(percent));
171        }
172
173        /**
174         * Adds a sub-task to the task.
175         *
176         * @param subtask  the subtask (<code>null</code> not permitted).
177         */
178        public void addSubtask(Task subtask) {
179            if (subtask == null) {
180                throw new IllegalArgumentException("Null 'subtask' argument.");
181            }
182            this.subtasks.add(subtask);
183        }
184
185        /**
186         * Removes a sub-task from the task.
187         *
188         * @param subtask  the subtask.
189         */
190        public void removeSubtask(Task subtask) {
191            this.subtasks.remove(subtask);
192        }
193
194        /**
195         * Returns the sub-task count.
196         *
197         * @return The sub-task count.
198         */
199        public int getSubtaskCount() {
```

C:/jfreechart-1.0.2/jfreechart-1.0.2-modified/source/org/jfree/data/gantt/Task.java, Top line: 139

```
139         * @return The task duration (possibly <code>null</code>).
140         */
141        public TimePeriod getDuration() {
142            return this.duration;
143        }
144
145        /**
146         * Sets the task duration (actual or estimated).
147         *
148         * @param duration  the duration (<code>null</code> permitted).
149         */
150        public void setDuration(TimePeriod duration) {
151            this.duration = duration;
152        }
153
154        /**
155         * Returns the percentage complete for this task.
156         *
157         * @return The percentage complete (possibly <code>null</code>).
158         */
159        public Double getPercentComplete() {
160            return this.percentComplete;
161        }
162
163        /**
164         * Sets the percentage complete for the task.
165         *
166         * @param percent  the percentage (<code>null</code> permitted).
167         */
168        public void setPercentComplete(Double percent) {
169            this.percentComplete = percent;
170        }
171
172        /**
173         * Sets the percentage complete for the task.
174         *
175         * @param percent  the percentage.
176         */
177        public void setPercentComplete(double percent) {
178            setPercentComplete(new Double(percent));
179        }
180
181        /**
182         * Adds a sub-task to the task.
183         *
184         * @param subtask  the subtask (<code>null</code> not permitted).
185         */
186        public void addSubtask(Task subtask) {
187            if (subtask == null) {
188                throw new IllegalArgumentException("Null 'subtask' argument.");
189            }
190            this.subtasks.add(subtask);
191        }
192
193        /**
194         * Removes a sub-task from the task.
195         *
196         * @param subtask  the subtask.
197         */
198        public void removeSubtask(Task subtask) {
199            this.subtasks.remove(subtask);
200        }
201
202        /**
203         * Returns the sub-task count.
204         *
205         * @return The sub-task count.
206         */
207        public int getSubtaskCount() {
```

C:/jfreechart-1.0.2/jfreechart-1.0.2/source/org/jfree/data/gantt/Task.java, Top line: 200

```
200         return this.subtasks.size();
201     }
202
203     /**
204      * Returns a sub-task.
205      *
206      * @param index  the index.
207      *
208      * @return The sub-task.
209      */
210     public Task getSubtask(int index) {
211         return (Task) this.subtasks.get(index);
212     }
213
214     /**
215      * Tests this object for equality with an arbitrary object.
216      *
217      * @param object  the other object (<code>null</code> permitted).
218      *
219      * @return A boolean.
220      */
221     public boolean equals(Object object) {
222         if (object == this) {
223             return true;
224         }
225         if (!(object instanceof Task)) {
226             return false;
227         }
228         Task that = (Task) object;
229         if (!ObjectUtilities.equal(this.description, that.description)) {
230             return false;
231         }
232         if (!ObjectUtilities.equal(this.duration, that.duration)) {
```

C:/jfreechart-1.0.2/jfreechart-1.0.2-modified/source/org/jfree/data/gantt/Task.java, Top line: 208

```
208         return this.subtasks.size();
209     }
210
211     /**
212      * Returns a sub-task.
213      *
214      * @param index  the index.
215      *
216      * @return The sub-task.
217      */
218     public Task getSubtask(int index) {
219         return (Task) this.subtasks.get(index);
220     }
221
222     /**
223      * Returns if the task is a summary task or not.
224      *
225      * @return A summary task boolean (never <code>null</code>).
226      */
227     public boolean isSummaryTask() {
228         return this.summaryTask;
229     }
230
231     /**
232      * Sets if the task is a summary task or not.
233      *
234      * @param summaryTask a summary task boolean.
235      */
236     public void setSummaryTask(boolean summaryTask) {
237         this.summaryTask = summaryTask;
238     }
239
240     /**
241      * Returns if the task is a milestone or not.
242      *
243      * @return A milestone boolean (never <code>null</code>).
244      */
245     public boolean isMilestone() {
246         return this.milestone;
247     }
248
249     /**
250      * Sets if the task is a milestone or not.
251      *
252      * @param milestone a milestone boolean.
253      */
254     public void setMilestone(boolean milestone) {
255         this.milestone = milestone;
256     }
257
258     /**
259      * Tests this object for equality with an arbitrary object.
260      *
261      * @param object  the other object (<code>null</code> permitted).
262      *
263      * @return A boolean.
264      */
265     public boolean equals(Object object) {
266         if (object == this) {
267             return true;
268         }
269         if (!(object instanceof Task)) {
270             return false;
271         }
272         Task that = (Task) object;
273         if (!ObjectUtilities.equal(this.description, that.description)) {
274             return false;
275         }
276         if (!ObjectUtilities.equal(this.duration, that.duration)) {
```

C:/jfreechart-1.0.2/jfreechart-1.0.2/source/org/jfree/data/gantt/Task.java, Top line: 233

```
233             return false;
234         }
235         if (!ObjectUtilities.equal(this.percentComplete,
236                 that.percentComplete)) {
237             return false;
238         }
239         if (!ObjectUtilities.equal(this.subtasks, that.subtasks)) {
240             return false;
241         }
242         return true;
243     }
244
245     /**
246      * Returns a clone of the task.
247      *
248      * @return A clone.
249      *
250      * @throws CloneNotSupportedException  never thrown by this class, but
251      *         subclasses may not support cloning.
252      */
253     public Object clone() throws CloneNotSupportedException {
254         Task clone = (Task) super.clone();
255         return clone;
256     }
257
258 }
259
```

C:/jfreechart-1.0.2/jfreechart-1.0.2-modified/source/org/jfree/data/gantt/Task.java, Top line: 277

```
277             return false;
278         }
279         if (!ObjectUtilities.equal(this.percentComplete,
280                 that.percentComplete)) {
281             return false;
282         }
283         if (!ObjectUtilities.equal(this.subtasks, that.subtasks)) {
284             return false;
285         }
286         return true;
287     }
288
289     /**
290      * Returns a clone of the task.
291      *
292      * @return A clone.
293      *
294      * @throws CloneNotSupportedException  never thrown by this class, but
295      *         subclasses may not support cloning.
296      */
297     public Object clone() throws CloneNotSupportedException {
298         Task clone = (Task) super.clone();
299         return clone;
300     }
301
302 }
303
```