GoF Rodrigo Jacob

* Iterator ([ExporterToMsProjectFile.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.impex.msproject2/src/main/java/biz/ganttproject/impex/msproject2/ExporterToMsProjectFile.java))

O método (getSelectedFormatExtension) itera sobre os elementos de uma coleção.

private String getSelectedFormatExtension() {

    for (int i = 0; i < FILE\_FORMAT\_IDS.length; i++) {

      if (myFileFormat.equals(FILE\_FORMAT\_IDS[i])) {

        return FILE\_EXTENSIONS[i];

      }

    }

    throw new IllegalStateException("Selected format=" + myFileFormat + " has not been found in known formats:"

        + Arrays.asList(FILE\_FORMAT\_IDS));

  }

* Facade ([ProjectFileExporter.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.impex.msproject2/src/main/java/biz/ganttproject/impex/msproject2/ProjectFileExporter.java))

O método (convertFinishTime) possui variáveis que representam instâncias de partes mais complexas dos sistemas, assim simplificando o seu uso.

private Date convertFinishTime(Date gpFinishDate) {

    Calendar c = (Calendar) Calendar.getInstance().clone();

    c.setTime(gpFinishDate);

    c.add(Calendar.DAY\_OF\_YEAR, -1);

    Date finishTime = myOutputProject.getDefaultCalendar().getFinishTime(c.getTime());

    if (finishTime != null) {

    c.set(Calendar.HOUR, finishTime.getHours());

    c.set(Calendar.MINUTE, finishTime.getMinutes());

    }

    return c.getTime();

  }

* Builder ([Canvas.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/canvas/Canvas.java))

A classe shape não tem construtor, as suas variáveis são inicializadas por partes (com setters).

public static class Shape {

    private Color myBackgroundColor;

    private Color myForegroundColor;

    private String myStyleName;

    private Object myModelObject;

    private boolean isVisible = true;

    private LinkedHashSet<String> myStyles;

    private Float myOpacity = null;

    private final Map<String, String> attributes = new HashMap<>();

    private LinkedHashSet<String> getStyles() {

      if (myStyles == null) {

        myStyles = new LinkedHashSet<String>();

      }

      return myStyles;

    }

    public void addStyle(String style) {

      getStyles().add(style);

    }

    public boolean hasStyle(String style) {

      return getStyles().contains(style);

    }

    public void setStyle(String styleName) {

      myStyleName = styleName;

    }

    public String getStyle() {

      return myStyleName;

    }

    public Color getBackgroundColor() {

      return myBackgroundColor;

    }

    public void setBackgroundColor(Color myBackgroundColor) {

      this.myBackgroundColor = myBackgroundColor;

    }

    public Color getForegroundColor() {

      return myForegroundColor;

    }

    public void setForegroundColor(Color myForegroundColor) {

      this.myForegroundColor = myForegroundColor;

    }

    public Object getModelObject() {

      return myModelObject;

    }

    public void setModelObject(Object modelObject) {

      myModelObject = modelObject;

    }

    public boolean isVisible() {

      return isVisible;

    }

    public void setVisible(boolean visible) {

      isVisible = visible;

    }

    public Float getOpacity() {

      return myOpacity;

    }

    public void setOpacity(float opacity) {

      myOpacity = opacity;

    }

    public Map<String, String> getAttributes() {

      return attributes;

    }

  }