ENGENHARIA DE SOFTWARE
2022/2023

Projeto - Fase 1

Autores:
45345 - Inês Carvalho
54622 - Tiago Coelho
58013 - Rodrigo Lopes
59208 - Simão Carrasco
60597 - Pedro Catarino

1. Identified Design Patterns

- 45345 - Inês Carvalho:
    - **Pattern 1 - Template Method Pattern:**
        - Illustraring code snippet:
            - On the GPCalenderBase class:
                - Examples of methods in common:

```
dbarashev
@Override
public void setName(String name) { myName = name; }

dbarashev
@Override
public void setID(String id) { myId = id; }
```

                - Examples of methods implemented differently:

```
1 usage  2 implementations  dbarashev
protected abstract List<GPCalendarActivity> getActivitiesBackward(Date startDate, TimeUnit timeUnit, long unitCount);

1 usage  2 implementations  dbarashev
protected abstract List<GPCalendarActivity> getActivitiesForward(Date startDate, TimeUnit timeUnit, long unitCount);
```

            - On the AlwaysWorkingTimeCalendarImpl class:

```
1 usage  dbarashev
@Override
protected List<GPCalendarActivity> getActivitiesForward(Date startDate, TimeUnit timeUnit, long unitCount) {
  Date activityStart = timeUnit.adjustLeft(startDate);
  Date activityEnd = activityStart;
  while (unitCount-- > 0) {
    activityEnd = timeUnit.adjustRight(activityEnd);
  }
  return Collections.singletonList((GPCalendarActivity) new CalendarActivityImpl(activityStart, activityEnd,  isWorkingTime: true));
}

1 usage  dbarashev
@Override
protected List<GPCalendarActivity> getActivitiesBackward(Date startDate, TimeUnit timeUnit, long unitCount) {
  Date activityEnd = timeUnit.adjustLeft(startDate);
  Date activityStart = activityEnd;
  while (unitCount-- > 0) {
    activityStart = timeUnit.jumpLeft(activityStart);
  }
  return Collections.singletonList((GPCalendarActivity) new CalendarActivityImpl(activityStart, activityEnd,  isWorkingTime: true));
}
```

- On theWeekendCalendarImpl:

```
@Override
protected List<GPCalendarActivity> getActivitiesForward(Date startDate, TimeUnit timeUnit, final long unitCount) {
  final List<GPCalendarActivity> result = new ArrayList<~>();
  dbarashev
  new ForwardTimeWalker( calendar: this, timeUnit) {
    2 usages
    long myUnitCount = unitCount;

    1 usage  dbarashev
    @Override
    protected void processWorkingTime(Date intervalStart, Date nextIntervalStart) {
      result.add(new CalendarActivityImpl(intervalStart, nextIntervalStart,  isWorkingTime: true));
      myUnitCount--;
    }

    1 usage  dbarashev
    @Override
    protected void processNonWorkingTime(Date intervalStart, Date workingIntervalStart) {
      result.add(new CalendarActivityImpl(intervalStart, workingIntervalStart,  isWorkingTime: false));
    }

    1 usage  dbarashev
    @Override
    protected boolean isMoving() { return myUnitCount > 0; }
  }.walk(startDate);
  return result;
}
```

```
1 usage  dbarashev
@Override
protected List<GPCalendarActivity> getActivitiesBackward(Date startDate, TimeUnit timeUnit, long unitCount) {
  List<GPCalendarActivity> result = new LinkedList<~>();
  Date unitStart = timeUnit.adjustLeft(startDate);
  while (unitCount > 0) {
    Date prevUnitStart = timeUnit.jumpLeft(unitStart);
    boolean isWeekendState = (getDayMask(prevUnitStart) & DayMask.WORKING) == 0;
    if (isWeekendState) {
      Date lastWorkingUnitStart = findClosest(prevUnitStart, timeUnit, MoveDirection.BACKWARD, DayType.WORKING);
      Date firstWeekendUnitStart = timeUnit.adjustRight(lastWorkingUnitStart);
      Date lastWeekendUnitEnd = unitStart;
      result.add( index: 0, new CalendarActivityImpl(firstWeekendUnitStart, lastWeekendUnitEnd,  isWorkingTime: false));
      unitStart = firstWeekendUnitStart;
    } else {
      result.add( index: 0, new CalendarActivityImpl(prevUnitStart, unitStart,  isWorkingTime: true));
      unitCount--;
      unitStart = prevUnitStart;
    }
  }
  return result;
}
```

- The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> calendar

- An explanation of the rationale for identifying this as a pattern instantiation: Both AlwaysWorkingTimeCalendarImpl and Weekend CalendarImpl classes extend GPCalendarBase abstract class. The subclasses both have common methods inherited from the template class, so it makes sense to identify this as an instance of a template method.

- **Pattern 2 - Factory Object:**

  - Illustraring code snippet:
    - On the OffsetBuilder interface:

```
protected Factory() {
}

6 usages  dbarashev
public Factory withTopUnit(TimeUnit topUnit) {
  myTopUnit = topUnit;
  return this;
}

6 usages  dbarashev
public Factory withBottomUnit(TimeUnit bottomUnit) {
  myBottomUnit = bottomUnit;
  return this;
}

dbarashev
public Factory withStartDate(Date startDate) {
  myStartDate = startDate;
  return this;
}
```

    - On OffsetBuilderImpl:

```
public static class FactoryImpl extends OffsetBuilder.Factory {
  dbarashev
  @Override
  public OffsetBuilder build() {
    preBuild();
    return new OffsetBuilderImpl( factory: this);
  }
}
```

  - The exact location on the codebase: Project -> Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> grid -> OffsetBuilder

- An explanation of the rationale for identifying this as a pattern instantiation: Hiding the creation of instances of a given type

- **Pattern 3 - Facade:**

  - Illustraring code snippet:

```
public abstract class AbstractSceneBuilder implements SceneBuilder {
public class BottomUnitSceneBuilder extends AbstractSceneBuilder
public class DayGridSceneBuilder extends AbstractSceneBuilder {
public class TimelineSceneBuilder extends AbstractSceneBuilder {
```

  - The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> chart -> scene -> AbstractSceneBuilder, BottomUnitSceneBuilder, DayGridSceneBuilder, TimelineSceneBuilder

  - An explanation of the rationale for identifying this as a pattern instantiation: Hide complexity behind an interface. Provides a unified interface to a complex subsystem

Reviews:

Reviewer's Name: Simão Carrasco
Pattern 1: The subclasses both have common methods inherited from the template class, so it makes sense to identify this as a code smell

Reviewer's Name: Pedro Catarino
The pattern was well identified and explained

- 54622 - Tiago Coelho:
    - **Pattern 1 - Facade:**
        - Illustraring code snippet:

```
public abstract class CachingTextFormatter {
public class DayTextFormatter extends CachingTextFormatter implements TimeFormatter {
public class MonthTextFormatter extends CachingTextFormatter implements TimeFormatter {
public class QuarterTextFormatter extends CachingTextFormatter implements TimeFormatter {
public class WeekTextFormatter extends CachingTextFormatter implements TimeFormatter {
public class YearTextFormatter extends CachingTextFormatter implements TimeFormatter {
```

        - The exact location on the codebase: Project -> biz.ganttproject.core [core] -> src -> main -> java -> bizganttproject -> core -> chart -> text
        - An explanation of the rationale for identifying this as a pattern instantiation: Structuring a system into subsystems helps reduce complexity
    - **Pattern 2 - Memento:**
        - Illustraring code snippet:

```
public Rhombus createRhombus(int leftx, int topy, int diagWidth, int diagHeight) {
  Rhombus rhombus = new Rhombus(leftx, topy, diagWidth, diagHeight);
  myRhombusIndex.put(rhombus, rhombus.getLeftX(), rhombus.getBottomY(), rhombus.getWidth(), rhombus.getHeight());
  return rhombus;
}
```

        - The exact location on the codebase: Project -> biz.ganttproject.core [core] -> src -> main -> java -> bizganttproject -> core -> chart -> chart -> canvas -> Canvas
        - An explanation of the rationale for identifying this as a pattern instantiation: This method needs to return an object.

- **Pattern 3 - Template:**
  - Illustraring code snippet:

```
public void build() {
  if (myRedlineOption.isChecked()) {
    renderLine(new Date(),  style: "timeline.today",  marginPx: 2, OffsetLookup.BY_END_DATE);
  }
  if (isProjectBoundariesOptionOn()) {
    renderLine(myInputApi.getProjectStart(),  style: "timeline.project_start",  marginPx: -2, OffsetLookup.BY_START_DATE);
    renderLine(myInputApi.getProjectEnd(),  style: "timeline.project_end",  marginPx: 2, OffsetLookup.BY_START_DATE);
  }
  renderNonWorkingDayColumns();
}

3 usages  dbarashev
private void renderLine(Date date, String style, int marginPx, OffsetLookup.ComparatorBy<Date> dateComparator) {
  final int topUnitHeight = myInputApi.getTopLineHeight();
  OffsetLookup lookup = new OffsetLookup();
  int todayOffsetIdx = lookup.lookupOffsetBy(date, myInputApi.getAtomUnitOffsets(), dateComparator);
  if (todayOffsetIdx < 0) {
    todayOffsetIdx = -todayOffsetIdx - 1;
  }
  Offset yesterdayOffset = todayOffsetIdx == 0 ? null : myInputApi.getAtomUnitOffsets().get(
      todayOffsetIdx - 1);
  if (yesterdayOffset == null) {
    return;
  }
  int yesterdayEndPixel = yesterdayOffset.getOffsetPixels();
  Line line = getCanvas().createLine( startx: yesterdayEndPixel + marginPx,  starty: topUnitHeight * 2,
      finishx: yesterdayEndPixel + marginPx,  finishy: getHeight() + topUnitHeight * 2);
  line.setStyle(style);
}
```

```
private void renderNonWorkingDayColumns() {
  List<Offset> defaultOffsets = myInputApi.getAtomUnitOffsets();
  int curX = defaultOffsets.get(0).getOffsetPixels();
  if (curX > 0) {
    curX = 0;
  }
  for (final Offset offset : defaultOffsets) {
    int dayMask = offset.getDayMask();
    CalendarEvent event = myInputApi.getEvent(offset.getOffsetStart());
    final int _curX = curX;
    dbarashev
    Runnable r = new Runnable() {
      dbarashev
      @Override
      public void run() {
        // Create a holiday/weekend day bar in the main area
        renderNonWorkingDay(_curX, offset);
        // And expand it to the timeline area.
        Rectangle r = myTimelineCanvas.createRectangle(_curX, getLineTopPosition(),
            width: offset.getOffsetPixels() - _curX,  height: getLineBottomPosition() - getLineTopPosition());
        applyRectangleStyle(r, offset);
      }
    };
    if ((dayMask & (DayMask.WEEKEND)) != 0) {
      // We render weekends always. If there is a colored event its color will be applied
      // in applyRectangleStyle because getholidaycolor returns non-null
      r.run();
    } else if (event != null) {
      // It is not a weekends but it is an event
      // Holidays should always be painted, but neutral and working days should not unless
      // they have a custom color
      if (event.getType() == CalendarEvent.Type.HOLIDAY || event.getColor() != null) {
        r.run();
```

```
private void renderNonWorkingDay(int curX, Offset curOffset) {
  Canvas.Rectangle r = getCanvas().createRectangle(curX, getLineBottomPosition(),
      width: curOffset.getOffsetPixels() - curX, getHeight());
  applyRectangleStyle(r, curOffset);
}

2 usages  dbarashev
private void applyRectangleStyle(Rectangle r, Offset offset) {
  Color customColor = myInputApi.getHolidayColor(offset.getOffsetStart());
  if (customColor != null) {
    r.setBackgroundColor(customColor);
    r.setOpacity(1.0f);
  }
  if ((offset.getDayMask() & DayMask.HOLIDAY) == DayMask.HOLIDAY) {
    r.setStyle("calendar.holiday");
    return;
  }
  if ((offset.getDayMask() & DayMask.WEEKEND) == DayMask.WEEKEND) {
    r.setStyle("calendar.weekend");
    return;
  }
}
```

- The exact location on the codebase: Project -> biz.ganttproject.core [core] -> src -> main -> java -> bizganttproject -> core -> chart -> scene -> gantt -> DayGridSceneBuilder

- An explanation of the rationale for identifying this as a pattern instantiation: Defines an algorithm`s steps.

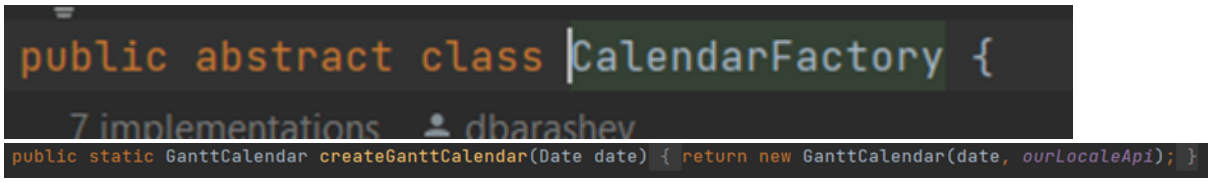
Reviews:

Reviewer's Name: Simão Carrasco
pattern 2: It's not clear how this method allows access to a previous state of an object

Reviewer's Name: Inês Carvalho
Pattern 3: Explanation seems very generic for what the pattern does

- 58013 - Rodrigo Lopes:
    - **Pattern 1 - <pattern name>:**
        - Illustraring code snippet:
        - The exact location on the codebase:
        - An explanation of the rationale for identifying this as a pattern instantiation:
    - **Pattern 2 - <pattern name>:**
        - Illustraring code snippet:
        - The exact location on the codebase:
        - An explanation of the rationale for identifying this as a pattern instantiation:
    - **Pattern 3 - <pattern name>:**
        - Illustraring code snippet:
        - The exact location on the codebase:
        - An explanation of the rationale for identifying this as a pattern instantiation:

- 59208 - Simão Carrasco:
    - **Pattern 1 - Factory Method Pattern:**
        - Illustraring code snippet:



        - The exact location on the codebase:ganttproject\biz.ganttproject.core\src\main\java\biz\ganttproject\core\time\CalendarFactory.java

        - An explanation of the rationale for identifying this as a pattern instantiation: The class CalendarFactory takes the responsibility of instancing the GanttCalendar from its respective class, to this factory class.

    - **Pattern 2 - Template Method Pattern:**
        - Illustraring code snippet:

```
public abstract class CachingTextFormatter {
  3 usages
  private final HashMap<Date, TimeUnitText[]> myTextCache = new HashMap<~>();
  2 usages
  private LocaleApi myLocale;

  5 usages  dbarashev
  protected CachingTextFormatter() {
  }
```

```
public class WeekTextFormatter extends CachingTextFormatter implements TimeFormatter {
  5 usages
public class YearTextFormatter extends CachingTextFormatter implements TimeFormatter {
```

```
protected abstract TimeUnitText[] createTimeUnitText(Date adjustedLeft);
```

```java
@Override
protected TimeUnitText[] createTimeUnitText(Date startDate) {
  myCalendar.setTime(startDate);
  // Integer yearNo = new Integer(myCalendar.get(Calendar.YEAR));
  // String shortText = MessageFormat.format("{0}", new Object[]
  // {yearNo});
  String shortText = MessageFormat.format( pattern: "{0,date,yyyy}", new Object[] { myCalendar.getTime() });
  return new TimeUnitText[] { new TimeUnitText(shortText) };
}
```

- The exact location on the codebase:ganttproject\biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\text\CachingTextFormatter.java

- An explanation of the rationale for identifying this as a pattern instantiation:The classes WeekTextFormatter and YearTextFormatter although different share a lot of common functionalities,thus making it useful to have a template method.

- **Pattern 3 - Facade Pattern:**

  - Illustraring code snippet:

```java
public TimeFormatters(LocaleApi localeApi) {
  Map<String, TimeFormatter> commonFormatters = new HashMap<>();

  commonFormatters.put(GPTimeUnitStack.DAY.getName(), new DayTextFormatter());
  commonFormatters.put(GPTimeUnitStack.QUARTER.getName(), new QuarterTextFormatter());
  commonFormatters.put(GPTimeUnitStack.YEAR.getName(), new YearTextFormatter());

  ourUpperFormatters.putAll(commonFormatters);
  ourUpperFormatters.put(GPTimeUnitStack.MONTH.getName(), new MonthTextFormatter(localeApi, longPattern: "MMMM yyyy", mediumPattern: "MMM''yyy
  ourUpperFormatters.put(GPTimeUnitStack.WEEK.getName(), new WeekTextFormatter());

  ourLowerFormatters.putAll(commonFormatters);
  ourLowerFormatters.put(GPTimeUnitStack.MONTH.getName(), new MonthTextFormatter(localeApi, longPattern: "MMMM", mediumPattern: "MMM", shortPatte
  ourLowerFormatters.put(GPTimeUnitStack.WEEK.getName(), new WeekTextFormatter());
  setLocaleApi(localeApi);
}
```

```java
public TimeFormatter getFormatter(TimeUnit timeUnit, TimeUnitText.Position position) {
  TimeFormatter result = DEFAULT_TIME_FORMATTER;
  switch (position) {
  case UPPER_LINE:
    result = ourUpperFormatters.get(timeUnit.getName());
    break;
  case LOWER_LINE:
    result = ourLowerFormatters.get(timeUnit.getName());
    break;
  }
  return result;
}
```

- The exact location on the codebase: ganttproject\biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\text\TimeFormatters.java

- An explanation of the rationale for identifying this as a pattern instantiation: Acts as a point of entry to all the different text formatter classes

- 60597 - Pedro Catarino:
  - **Pattern 1 - Template method:**
    - Illustraring code snippet:

On *ChartComponentBase*:

```
@Override
public IGanttProject getProject() { return myProject; }

@Override
public Date getEndDate() { return getImplementation().getEndDate(); }
```

```
protected abstract ChartModelBase getChartModel();

protected abstract MouseListener getMouseListener();
```

On *GanttGraphicArea*:

```
public ChartModelImpl getMyChartModel() { return myChartModel; }
```

On *ResourceLoadGraphicArea:*

```
protected ChartModelBase getChartModel() { return myChartModel; }
```

- The exact location on the codebase:

  Project > ganttproject > src > main > java > net > sourceforge > ganttproject > ChartComponentBase, GanttGraphicArea, ResourceLoadGraphicArea

- An explanation of the rationale for identifying this as a pattern instantiation:

  GanttGraphicArea and *ResourceLoadGraphicArea* have a similar implementation that they share with the abstract class *ChartComponentBase,* the differences in their implementation are resolved in each of the subclasses.

- **Pattern 2 - Singleton:**

  - Illustraring code snippet:

Instantiation:

```
private GanttCalendar myShiftedValue;
```

Lazy construction:

```
public GanttCalendar getDisplayValue() {
  if (myShiftedValue == null) {
    myShiftedValue = CalendarFactory.createGanttCalendar(GPTimeUnitStack.DAY.jumpLeft(getTime()));
  }
  return myShiftedValue;
}
}
```

  - The exact location on the codebase:

    Project > biz.ganttproject.core > src > main > java > biz > ganttproject > core > time > GanttCalendar

  - An explanation of the rationale for identifying this as a pattern instantiation:

    There is only one instance of this class and it is responsible for itself.

Reviewer's Name: Simão Carrasco
Pattern 2: There is always only one instance of the class, so it makes sense to identify it as a singleton.

- **Pattern 3 - Memento:**
  - Illustraring code snippet:

```
  myDocumentBefore = saveFile();
  try {
    projectDatabaseTxn = myManager.getProjectDatabase().startTransaction(localizedName);
    editImpl.run();
    projectDatabaseTxn.commit();
  } catch (ProjectDatabaseException ex) {
    GPLogger.log(ex);
  }
  myDocumentAfter = saveFile();
}
```

  - The exact location on the codebase:

    Project > ganttproject > src > main > java > net > sourceforge > ganttproject > undo > UndoableEditImpl

  - An explanation of the rationale for identifying this as a pattern instantiation:

    The pattern is clearly used to allow for an Undo option to get a previous version of the document.

Reviewer's Name: Inês Carvalho

Looks good to me

2. Identified Code Smells

- 45345 - Inês Carvalho:
    - **Code Smell 1 - Dead Code:**
        - Illustrating code snippet:

```
public interface GPCalendarListener {
  1 usage   dbarashev
  void onCalendarChange();
}
```

        - The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> calendar -> GPCalendarListener
        - An explanation of the rationale for identifying this code smell: This interface has only one void method. Classes do not implement this interface, yet the method has 4 usages calls on the WeekendCalendarImpl fileCalendarChanged().
        - A refactoring proposal Use a variable on the class as a flag to register the change and then call a method that implements the change. In this case it would be the notification:

- **Code Smell 2 - Primitive Obsession:**
  - Illustrating code snippet:

```
public class DummySpatialIndex<T> implements SpatialIndex<T>{

  3 usages  dbarashev
  private static class Rect<T> {
    2 usages
    final T myObject;
    4 usages
    final int myBottomY;
    3 usages
    private int myWidth;
    2 usages
    private int myHeight;
    4 usages
    private int myLeftX;

    1 usage  dbarashev
    Rect(T object, int leftX, int bottomY, int width, int height) {
      myObject = object;
      myBottomY = bottomY;
      myLeftX = leftX;
      myWidth = width;
      myHeight = height;
    }
```

  - The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> chart -> DummySpatialIndex.java
  - An explanation of the rationale for identifying this code smell: Unnecessary use of a lot of primitive types, thus it makes sense to identify this as a code smell
  - A refactoring proposal:The variables that represent the coordinates could be changed to an abstract type such as a tuple Point

- **Code Smell 3 - Shotgun Surgery:**
  - Illustrating code snippet:
    - On RectangleRenderer class:

```
public RectangleRenderer(Properties props) { myProperties = props; }

dbarashev
public void setGraphics(Graphics2D graphics) { myGraphics = graphics; }
```

```
public boolean render(Canvas.Rectangle rect) {
  Graphics2D g = (Graphics2D) myGraphics.create();
  Style style = Style.getStyle(myProperties, rect.getStyle());

  if (style.getVisibility(rect) == Style.Visibility.HIDDEN) {
    return false;
  }
  Style.Color background = style.getBackgroundColor(rect);
  if (background != null) {
    g.setColor(background.get());
  }
  Paint paint = style.getBackgroundPaint(rect);
  if (paint != null) {
    g.setPaint(paint);
  }
  Float opacity = style.getOpacity(rect);
  if (opacity != null) {
    g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, opacity.floatValue()));
  }
  Style.Padding padding = style.getPadding();
  if (style.getBackgroundImage() != null) {
    g.drawImage(style.getBackgroundImage(), x: rect.getLeftX() + padding.getLeft(), y: rect.getTopY() + padding.
  } else {
    g.fillRect( x: rect.getLeftX() + padding.getLeft(), y: rect.getTopY() + padding.getTop(),
        width: rect.getWidth() - (padding.getLeft() + padding.getRight()), height: rect.getHeight() - (padding.getT
  }
  Style.Borders border = style.getBorder(rect);
```

    - On SummaryTaskRenderer class:

```
1 usage   Dmitry Kazakov
public SummaryTaskRenderer(Properties props) { myProperties = props; }

Dmitry Kazakov
public void setGraphics(Graphics2D graphics) { myGraphics = graphics; }
```

```
public void render(Canvas.Rectangle rect) {
  Graphics2D g = (Graphics2D) myGraphics.create();
  Style style = Style.getStyle(myProperties, rect.getStyle());

  Style.Color background = style.getBackgroundColor(rect);
  if (background != null) {
    g.setColor(background.get());
  }
  Style.Padding padding = style.getPadding();
  g.fillRect( x: rect.getLeftX() + padding.getLeft(),  y: rect.getTopY() + padding.getTop(),
          width: rect.getWidth() - (padding.getLeft() + padding.getRight()),  height: rect.getHeight() - (padding.

  if (rect.hasStyle("task.summary.open")) {
    g.fillPolygon(
        new int[] { rect.getLeftX(), rect.getLeftX() + rect.getHeight(), rect.getLeftX() },
        new int[] { rect.getTopY(), rect.getTopY(), rect.getBottomY() },
         nPoints: 3
    );
  }
  if (rect.hasStyle("task.summary.close")) {
    g.fillPolygon(
        new int[] { rect.getRightX(), rect.getRightX() - rect.getHeight(), rect.getRightX() },
        new int[] { rect.getTopY(), rect.getTopY(), rect.getBottomY() },
         nPoints: 3
    );
  }
}
```

- The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> chart -> render -> LineRenderer.java, PolygonRenderer.java, Rectangle Renderer, SummaryTaskRenderer

- An explanation of the rationale for identifying this code smell: LineRenderer, PolygonRenderer, RectangleRenderer and SummaryTaskRenderer have some similar methods and methods with the same name implemented differently. If an update is made change needs to happen in multiple places

- A refactoring proposal: This appears to be an abstraction. As a possible refactoring I would make an AbstractClass called Renderer and would implement the similarities there, extend the other classes and implement the differences there. That way when an update is to be made that is equal to all types of renderer, then change only happens in one place

Reviews:

Reviewer's Name: Simão Carrasco
Unnecessary use of a lot of primitives types, thus it makes sense to identify this as a code smell

Reviewer's Name: Pedro Catarino
Code Smell 3: Everything seems good.

Reviewer´s Name: Tiago Coelho
The class appears to be an abstract class. So we should implement new classes that extend the main class of shape, and those new classes that specify the different shapes. With this, the code would be better organized and more readable for programmers who read it.

- 54622 - Tiago Coelho:
    - **Code Smell 1 - Speculative Generality:**
        - Illustrating code snippet:

```
public class TimeUnitDateFrameableImpl extends TimeUnitImpl {
```

```
public class TimeUnitFunctionOfDateImpl extends TimeUnitDateFrameableImpl implements TimeUnitFunctionOfDate {
```

        - The exact location on the codebase: Project -> biz.ganttproject.core [core] -> src -> main -> java -> bizganttproject -> core -> time -> impl
        - An explanation of the rationale for identifying this code smell: The superclass "TimeUnitDateFrameableImpl" was created which, at the moment, is not necessary for the realization of this project, but it may be useful in the future.
        - A refactoring proposal: Implement the desired class without extending the superclass.

- **Code Smell 2 - Dead Code:**
  - Illustrating code snippet:

```
public boolean isADayOff(GanttCalendar date) {
  return (date.equals(myStart) || date.equals(myFinish) || (date.before(myFinish) && date.after(myStart)));
}

dbarashev
public boolean isADayOff(Date date) {
  return (date.equals(myStart.getTime()) || date.equals(myFinish.getTime()) || (date.before(myFinish.getTime()) && date.a
}

dbarashev
public int isADayOffInWeek(Date date) {
  GanttCalendar start = myStart.clone();
  GanttCalendar finish = myFinish.clone();
  for (int i = 0; i < 7; i++) {
    start.add(Calendar.DATE,  amount: -1);
    finish.add(Calendar.DATE,  amount: -1);
    if (date.equals(start.getTime()) || date.equals(finish.getTime())
        || (date.before(finish.getTime()) && date.after(start.getTime())))
      return i + 1;
  }
  return -1;
}
```

  - The exact location on the codebase:Project -> biz.ganttproject.core [core] -> src -> main -> java -> bizganttproject -> core -> calendar -> walker -> GanttDaysOff
  - An explanation of the rationale for identifying this code smell: Methods ´isDayOff (GanttCalendar date)´, ´isDayOff (Date date)´ and isDayOfffInWeek (Date date)´ are never used.
  - A refactoring proposal: If these methods are really never used, and are not used in the future, we can delete them.

- **Code Smell 3 - Data Clump:**
  - Illustrating code snippet:

```
List<GPCalendarActivity> activities = getActivities(input, shift);
if (activities.isEmpty()) {
  throw new RuntimeException("FIXME: Failed to compute calendar activities in time period=" + shift
      + " starting from " + input);
}
```

  - The exact location on the codebase: Project -> biz.ganttproject.core [core] -> src -> main -> java -> bizganttproject -> core -> calendar -> walker -> GPCalendarBase
  - An explanation of the rationale for identifying this code smell: The exception “RunTimeException” is handled in the middle of the method.

- A refactoring proposal: This exception could be thrown in this class, but could be handled in a diferent class with all other exceptions in the project.

Reviews:

Reviewer's Name: Simão Carrasco
Code smell 2: A chunk of code exists but no longer useful, so it makes sense to identify this as a code smell

Reviewer's Name: Inês Carvalho
Looks good to me

- 58013 - Rodrigo Lopes:
  - **Code Smell 1 - Long method:**
    - Illustrating code snippet:

```
public static @NotNull Runnable createAutosaveCleanup() {
  long now = CalendarFactory.newCalendar().getTimeInMillis();
  final File tempDir = getTempDir();
  final long cutoff;
  try {
    File optionsFile = GanttOptions.getOptionsFile();
    if (!optionsFile.exists()) {
      return () -> {};
    }
    BasicFileAttributes attrs = Files.readAttributes(optionsFile.toPath(), BasicFileAttributes.class);
    FileTime accessTime = attrs.lastAccessTime();
    FileTime modifyTime = attrs.lastModifiedTime();
    long lastFileTime = Math.max(accessTime.toMillis(), modifyTime.toMillis());
    cutoff = Math.min(lastFileTime, now);
  } catch (IOException e) {
    GPLogger.log(e);
    return () -> {};
  }
  return new Runnable() {
    @Override
    public void run() {
      GPLogger.log("Deleting old auto-save files");
      deleteAutosaves();
    }

    private void deleteAutosaves() {
      // Let's find autosaves created before launch of this GP instance
      File[] previousAutosaves = tempDir.listFiles(new FileFilter() {
```

    - The exact location on the codebase: /Users/rodrigo/Desktop/ganttproject-master/ganttproject/src/main/java/net/sourceforge/ganttproject/document/DocumentCreator.java

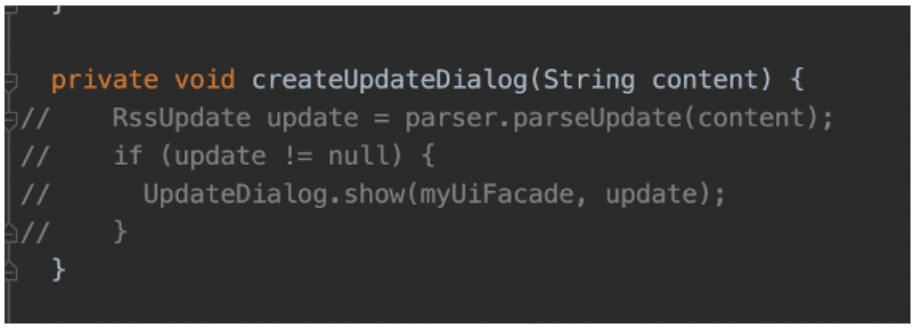
- An explanation of the rationale for identifying this code smell: The method is too long and hard to understand.
- A refactoring proposal: Split the long method into smaller methods.

- **Code Smell 2 - Clarification Comment:**
  - Illustrating code snippet:

```
Date lastCheck = myLastCheckOption.getValue();
if (lastCheck == null) {
  // It is the first time we run, just mark it. We want to suggest
  // subscribing to updates only to
  // those who runs GP at least twice.
  markLastCheck();
} else if (wasToday(lastCheck)) {
  // It is not the first run of GP but it was last run today and RSS
  // proposal has not been shown yet.
  // Add it to RSS button but don't promote it, wait until tomorrow.
  if (CheckOption.UNDEFINED == checkOption) {
    NotificationChannel.RSS.setDefaultNotification(myRssProposalNotification);
  }
} else {
  // So it is not the first time and even not the first day we start GP.
  // If no decision about subscribing, let's proactively suggest it,
  // otherwise
  // run check RSS.
```

  - The exact location on the codebase: /Users/rodrigo/Desktop/ganttproject-master/ganttproject/src/main/java/net/sourceforge/ganttproject/client/RssFeedChecker.java
  - An explanation of the rationale for identifying this code smell: Too many comments on the method
  - A refactoring proposal: Deleting the comments and leaving only the code

- **Code Smell 3 - Dead code:**
  - Illustrating code snippet:



```
  private void createUpdateDialog(String content) {
//    RssUpdate update = parser.parseUpdate(content);
//    if (update != null) {
//      UpdateDialog.show(myUiFacade, update);
//    }
  }
```

  - The exact location on the codebase: /Users/rodrigo/Desktop/ganttproject-master/ganttproject/src/main/java/net/sourceforge/ganttproject/client/RssFeedChecker.java
  - An explanation of the rationale for identifying this code smell: The method createUpdateDialog doesn't do anything
  - A refactoring proposal: Deleting the method and the places where it is used

- 59208 - Simão Carrasco:
    - **Code Smell 1 - Duplicated Code:**
        - Illustrating code snippet:

```
Object control = tableSkin.getSkinnable();
if (control instanceof TableView) {
  return ((TableView)control).resizeColumn((TableColumn)tc, delta);
} else if (control instanceof TreeTableView) {
  return ((TreeTableView)control).resizeColumn((TreeTableColumn)tc, delta);
}
```

```
public static boolean resizeColumn(TableViewSkinBase<?,?,?,?,?> tableSkin, Table
  if (!tc.isResizable()) return false;

  Object control = tableSkin.getSkinnable();
  if (control instanceof TableView) {
    return ((TableView)control).resizeColumn((TableColumn)tc, delta);
  } else if (control instanceof TreeTableView) {
    return ((TreeTableView)control).resizeColumn((TreeTableColumn)tc, delta);
  }
  return false;
}
```

```
public static BooleanProperty tableMenuButtonVisibleProperty(TableViewSkinBase<?,?,?,
  Object control = tableSkin.getSkinnable();
  if (control instanceof TableView) {
    return ((TableView)control).tableMenuButtonVisibleProperty();
  } else if (control instanceof TreeTableView) {
    return ((TreeTableView)control).tableMenuButtonVisibleProperty();
  }
  return null;
}
```

```
Object control = tableSkin.getSkinnable();
if (control instanceof TableView) {
  return ((TableView)control).tableMenuButtonVisibleProperty();
} else if (control instanceof TreeTableView) {
  return ((TreeTableView)control).tableMenuButtonVisibleProperty();
}
```

        - The exact location on the codebase:ganttproject\ganttproject\src\main\java\biz\ganttproject\lib\fx\treetable\TableSkinUtils.java

- An explanation of the rationale for identifying this code smell:
  Very similar chunks of code, with not a lot of differences, which are repeated multiple times across the class.

- A refactoring proposal: Create a new function that returns the referred chunk of code, and the arguments of this new function could be used to implement the small changes between the different implementations across the class.

Reviews:

Reviewer´s Name: Tiago Coelho
This code is duplicated and is used many times throughout the code. It can make the code confusing and make the programmer lose his mind. Should be created a new function, and whenever it is necessary to do this task, we call this new function.

- **Code Smell 2 - Data Clump:**

  - Illustrating code snippet:

```
private Rectangle(int leftx, int topy, int width, int height) { super( ...points: leftx, topy, leftx + width, topy + height); }
```

```
private Rhombus(int leftx, int topy, int diagWidth, int diagHeight) {
  super(
      ...points: leftx, topy + diagHeight / 2,
      leftx + diagWidth / 2, topy,
      leftx + diagWidth, topy + diagHeight / 2,
      leftx + diagWidth / 2, topy + diagHeight
  );
}
```

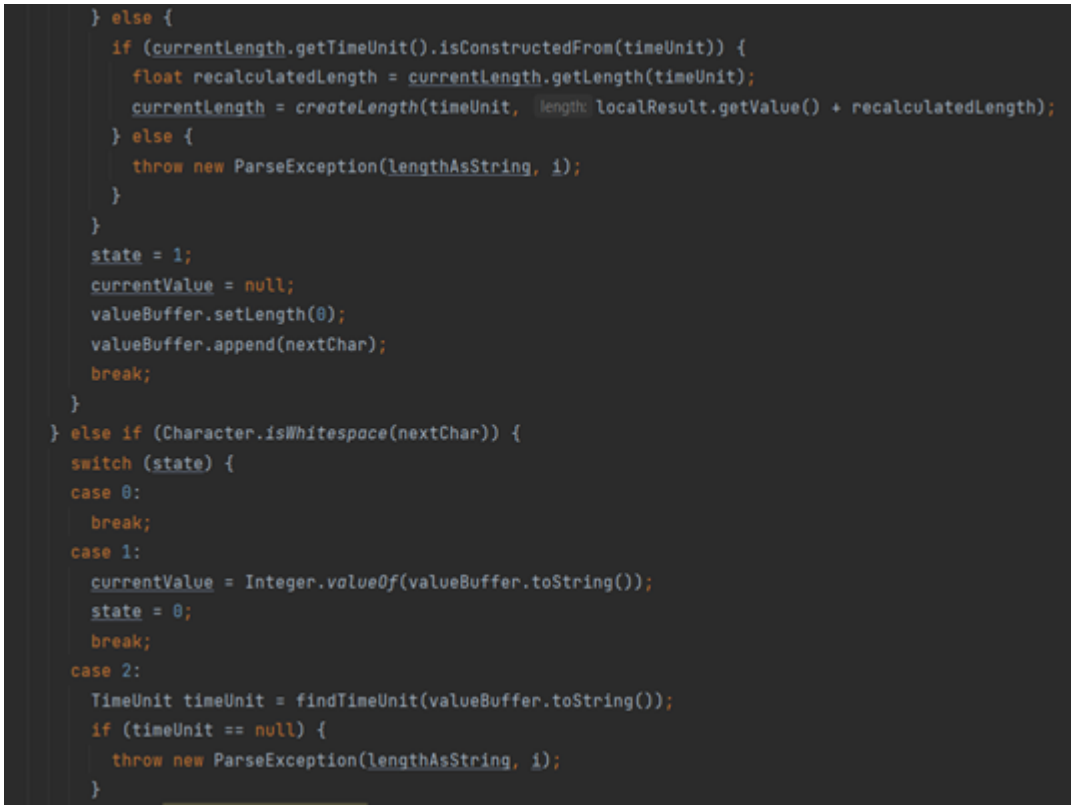
  - The exact location on the codebase:ganttproject\biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\canvas\Canvas.java

  - An explanation of the rationale for identifying this code smell:
    The same pair of arguments is used multiple times across the class

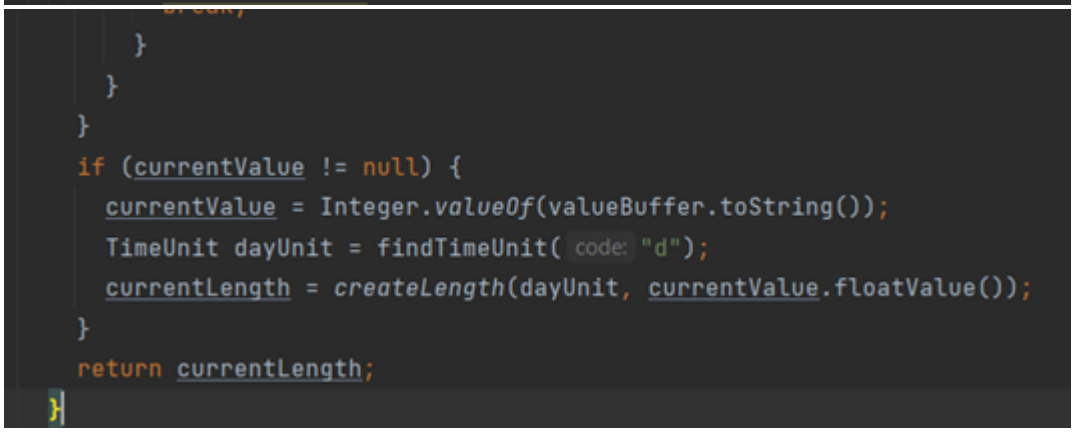
  - A refactoring proposal:
    Create a new object that encompasses this pair or primitives

- **Code Smell 3 - Long method:**
  - Illustrating code snippet:

```
public TimeDuration parseDuration(String lengthAsString) throws ParseException {
    int state = 0;
    StringBuffer valueBuffer = new StringBuffer();
    Integer currentValue = null;
    TimeDuration currentLength = null;
    lengthAsString += " ";
    for (int i = 0; i < lengthAsString.length(); i++) {
      char nextChar = lengthAsString.charAt(i);
      if (Character.isDigit(nextChar)) {
        switch (state) {
        case 0:
          if (currentValue != null) {
            throw new ParseException(lengthAsString, i);
          }
          state = 1;
          valueBuffer.setLength(0);
        case 1:
          valueBuffer.append(nextChar);
          break;
        case 2:
          TimeUnit timeUnit = findTimeUnit(valueBuffer.toString());
          if (timeUnit == null) {
            throw new ParseException(lengthAsString, i);
          }
          assert currentValue != null;
          TimeDuration localResult = createLength(timeUnit, currentValue.floatValue());
          if (currentLength == null) {
            currentLength = localResult;
          assert currentValue != null;
          TimeDuration localResult = createLength(timeUnit, currentValue.floatValue());
          if (currentLength == null) {
            currentLength = localResult;
          } else {
            if (currentLength.getTimeUnit().isConstructedFrom(timeUnit)) {
              float recalculatedLength = currentLength.getLength(timeUnit);
              currentLength = createLength(timeUnit, length: localResult.getValue() + recalculatedLength);
            } else {
              throw new ParseException(lengthAsString, i);
            }
          }
          state = 0;
          currentValue = null;
          break;
        }
      } else {
        switch (state) {
        case 1:
          currentValue = Integer.valueOf(valueBuffer.toString());
        case 0:
          if (currentValue == null) {
            throw new ParseException(lengthAsString, i);
          }
          state = 2;
          valueBuffer.setLength(0);
        case 2:
          valueBuffer.append(nextChar);
          break;
```



```
    } else {
      if (currentLength.getTimeUnit().isConstructedFrom(timeUnit)) {
        float recalculatedLength = currentLength.getLength(timeUnit);
        currentLength = createLength(timeUnit, length: localResult.getValue() + recalculatedLength);
      } else {
        throw new ParseException(lengthAsString, i);
      }
    }
    state = 1;
    currentValue = null;
    valueBuffer.setLength(0);
    valueBuffer.append(nextChar);
    break;
  }
} else if (Character.isWhitespace(nextChar)) {
  switch (state) {
  case 0:
    break;
  case 1:
    currentValue = Integer.valueOf(valueBuffer.toString());
    state = 0;
    break;
  case 2:
    TimeUnit timeUnit = findTimeUnit(valueBuffer.toString());
    if (timeUnit == null) {
      throw new ParseException(lengthAsString, i);
    }
```



```
      }
    }
  }
  if (currentValue != null) {
    currentValue = Integer.valueOf(valueBuffer.toString());
    TimeUnit dayUnit = findTimeUnit( code: "d");
    currentLength = createLength(dayUnit, currentValue.floatValue());
  }
  return currentLength;
}
```

- The exact location on the codebase:
  ganttproject\biz.ganttproject.core\src\main\java\biz\ganttproject\core\time\impl\GPTimeUnitStack.java

- An explanation of the rationale for identifying this code smell:
  Really long method, suggesting that some of its functionalities should be dispersed across different methods.

- A refactoring proposal: Divide the code across new methods, and by doing this increasing the legibility and structure of the code.

Reviews:

Reviewer`s Name: Tiago Coelho
This method is too long, so it would need to be better structured. It should start by implementing small functions that will be used throughout the code. Everytime it was necessary to do a task, the corresponding function would be called. By building several small functions like this, we would be able to build a more readable and organized code.

- 60597 - Pedro Catarino:
    - **Code Smell 1 - Dead code:**
        - Illustrating code snippet:

```
public void setMaxLength(int maxLength) {
  myMaxLength = maxLength;
}

public int getMaxLength() {
  return myMaxLength;
}
```

        - The exact location on the codebase:

          Project > biz.ganttproject.core > src > main > java > biz > ganttproject > core > chart > canvas > Canvas

        - An explanation of the rationale for identifying this code smell:

          The methods setMaxLength() and getMaxLength() are never used.

        - A refactoring proposal:

          Delete the unused methods as they are not needed.

- **Code Smell 2 - Data clump:**
  - Illustrating code snippet:

```
TimeUnit createTimeUnit(String name, TimeUnit atomUnit, int count) {
  TimeUnit result = new TimeUnitImpl(name, this, atomUnit);
  registerTimeUnit(result, count);
  return result;
}

public TimeUnit createDateFrameableTimeUnit(String name, TimeUnit atomUnit, int atomCount, DateFrameable framer) {
  TimeUnit result = new TimeUnitDateFrameableImpl(name, this, atomUnit, framer);
  registerTimeUnit(result, atomCount);
  return result;
}

public TimeUnitFunctionOfDate createTimeUnitFunctionOfDate(String name, TimeUnit atomUnit, DateFrameable framer) {
  TimeUnitFunctionOfDate result;
  result = new TimeUnitFunctionOfDateImpl(name, this, atomUnit, framer);
  registerTimeUnit(result,  atomCount: -1);
  return result;
}
```

  - The exact location on the codebase:

    Project > biz.ganttproject.core > src > main > java > biz > ganttproject > core > time > TimeUnitGraph

  - An explanation of the rationale for identifying this code smell:

    The parameters *name* and *atomUnit* are passed together often.

  - A refactoring proposal:

    Using an object that contains a name and atomUnit and pass that object as a parameter.

Reviews:

Reviewer's Name: Simão Carrasco

Code Smell 2: Repeated pairs of the same arguments on multiple functions, so it makes sense to identify this as a code smell

- **Code Smell 3 - Duplicate code:**
  - Illustrating code snippet:

```
public static CalendarEvent newEvent(Date date, boolean isRecurring, Type type, String title, Color color) {
  return new CalendarEvent(date, isRecurring, type, title, color);
}
```

  - The exact location on the codebase:

    Project > biz.ganttproject.core > src > main > java > biz > ganttproject > core > calendar > CalendarEvent

  - An explanation of the rationale for identifying this code smell:

    Redundant method.

  - A refactoring proposal:

    Remove the newEvent() method and just use the constructor to create new Events.

Reviewer's name: Inês Carvalho

Reviews: Looks good to me