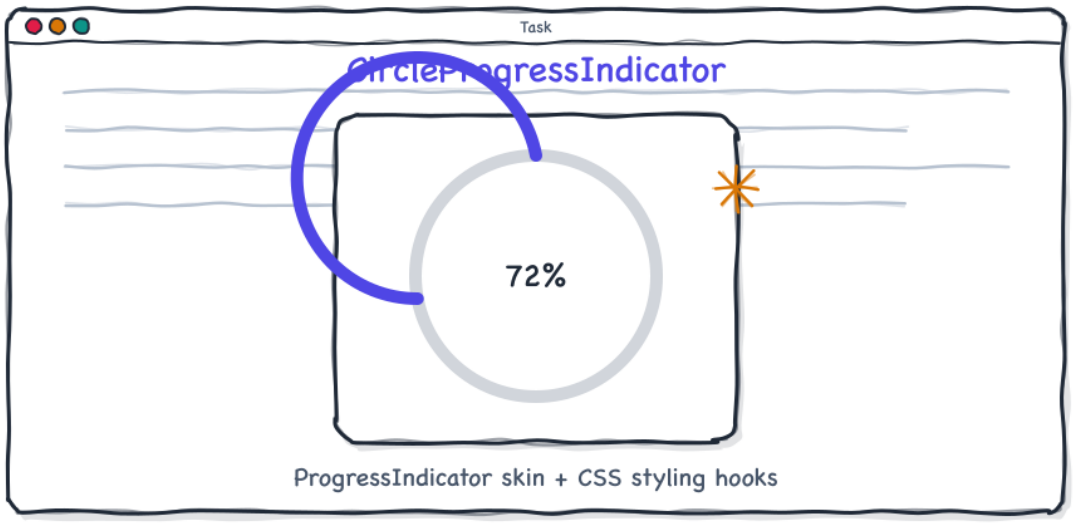
CircleProgressIndicator

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



A full circular progress indicator with a centered label.

CircleProgressIndicator is a concrete arc-based ProgressIndicator that draws a full circular track, a foreground progress arc and an optional centered text or graphic. It inherits the shared ArcProgressIndicator API and adds a startAngle property.

Table of Contents

1. Introduction

CircleProgressIndicator extends `ArcProgressIndicator`. It is the concrete choice for download progress, import jobs, synchronization and other tasks where compact circular feedback fits the surrounding layout.

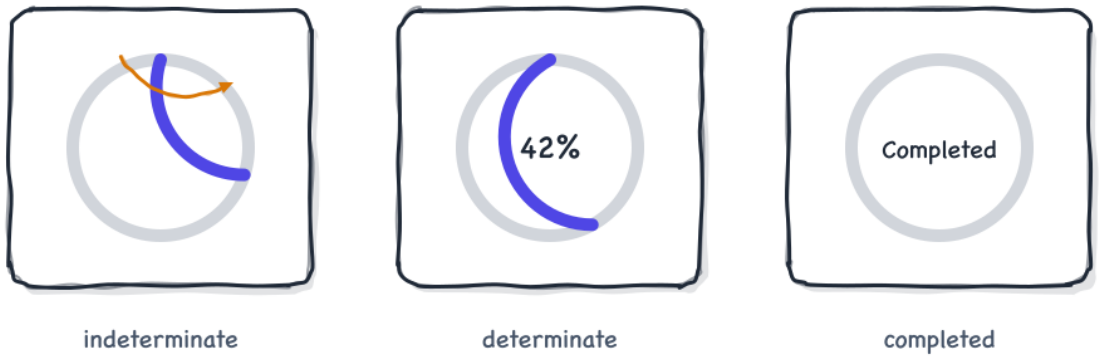
- Default constructor starts indeterminate.
- The `CircleProgressIndicator(double)` constructor sets the initial progress.
- Minimum size is set to 26 x 26 pixels.
- The root style classes are `arc-progress-indicator` and `circle-progress-indicator`.

```
CircleProgressIndicator indicator = new CircleProgressIndicator(0.0);
indicator.setProgress(0.35);
indicator.setStartAngle(90);
indicator.setStyleType(ArcProgressIndicator.StyleType.BOLD);
```

2. Getting started

Bind the inherited progress property to a task or service. Progress below zero makes the indicator indeterminate; progress from 0 to 1 draws that fraction of the circle.

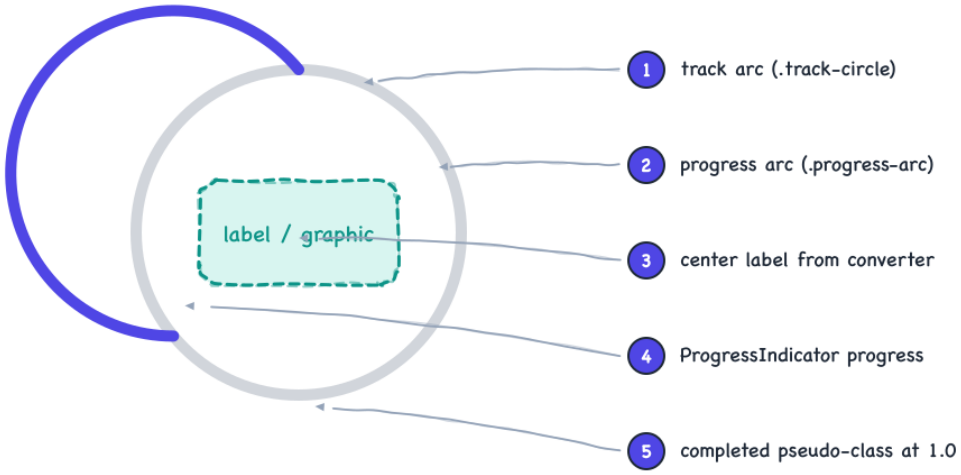
```
Service<Void> service = createService();
CircleProgressIndicator indicator = new CircleProgressIndicator();
indicator.progressProperty().bind(service.progressProperty());
service.start();
```



Indeterminate, determinate and completed circle states.

> **Tip.** Use `new CircleProgressIndicator(0.0)` when you want an empty determinate indicator instead of a running indeterminate animation.

3. Anatomy



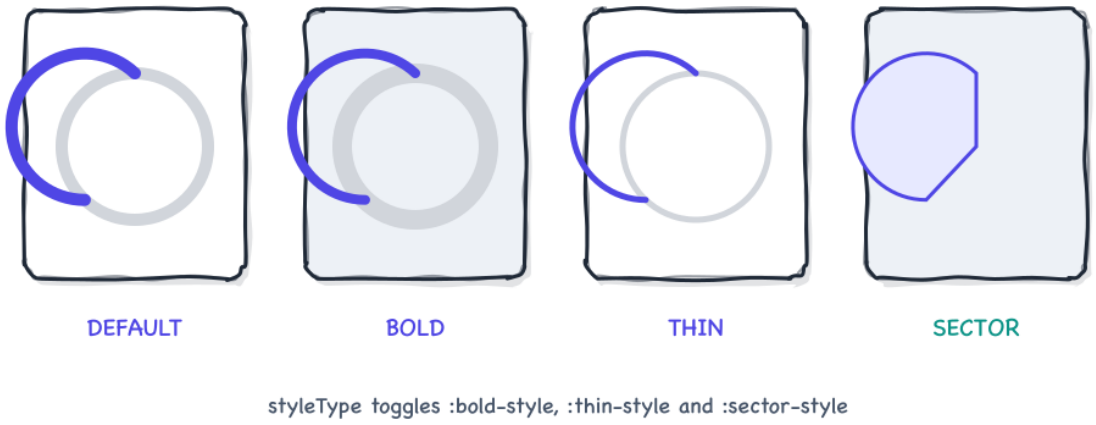
The parts rendered by CircleProgressIndicatorSkin.

Part	Node	Description
Track	Arc	Full 360 degree background track, length 360.
Progress	Arc	Foreground arc; length is -360 * progress for determinate values.
Label	Label	Centered in the circular interior and bound to converter / graphic.
Rotation	Rotate	Applied to the progress arc while indeterminate.

4. Control API

Property	Type	Default	Description
startAngle	DoubleProperty	90.0	Start angle of the progress arc, in degrees. 90 degrees puts the origin at the top of the circle.
converter	ObjectProperty<StringConverter<Double>>	percent converter	Inherited. Produces the centered label text.
graphic	ObjectProperty<Node>	null	Inherited. Custom graphic shown by the centered label.
progressArcType	ObjectProperty<ArcType>	OPEN	Inherited and styleable via -fx-progress-arc-type.
trackArcType	ObjectProperty<ArcType>	CHORD	Inherited and styleable via -fx-track-arc-type.
styleType	ObjectProperty<StyleType>	DEFAULT	Inherited. DEFAULT, BOLD, THIN or SECTOR.

5. Behaviour and states

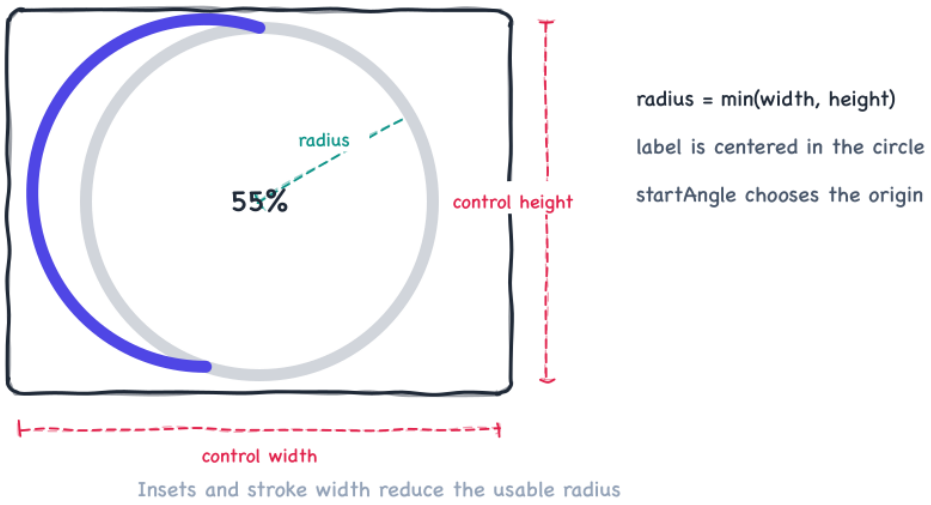


The built-in style types in the default CSS.

The skin creates its indeterminate timeline lazily. The timeline rotates the progress arc from 0 to 360 degrees over 1.5 seconds while the arc length grows from 45 to 180 and back to 45 degrees.

Progress	Label from default converter	Pseudo-classes
< 0	empty string	inherited indeterminate
0.0 ... 0.999	floored percent, e.g. 99.8 becomes 99%	inherited determinate
1.0	localized Completed	determinate + completed

6. Layout and sizing



The circular radius is derived from the smaller control dimension.

The radius binding uses `(min(width, height) - max(insets) - maxStrokeWidth) / 2`. The label gets the full inner diameter for both width and height, then is centered on the arc center.

7. Styling with CSS

Selector	Effect
`.circle-progress-indicator`	Root class loaded from circle-progress-indicator.css.
`.track-circle`	Light grey track in the concrete stylesheet.
`.progress-arc`	Uses -fx-accent unless overridden.
`:bold-style`	Track 10px, progress 5px.
`:thin-style`	Track and progress 1px.
`:sector-style`	Progress arc type ROUND and filled sector styling.
`:completed`	Additional hook for progress == 1.0.

CSS property	Type	Default
`-fx-style-type`	StyleType	DEFAULT
`-fx-progress-arc-type`	ArcType	OPEN
`-fx-track-arc-type`	ArcType	CHORD

```
.circle-progress-indicator {
    -fx-style-type: thin;
}

.circle-progress-indicator .progress-arc {
    -fx-stroke: -fx-accent;
}
```

8. Localization

CircleProgressIndicator inherits the arc-progress-indicator resource bundle.

Key	English text
`status.completed`	Completed
`accessible.text.loading`	loading
`accessible.text.percent`	{0} percent

9. Accessibility

The inherited constructor sets `AccessibleRole.PROGRESS_INDICATOR` and binds accessible text to progress. The automatic binding yields after application code calls `setAccessibleText`.

10. Recipes and demo

10.1 Start angle slider

```
Slider slider = new Slider(0, 360, 90);
indicator.startAngleProperty().bind(slider.valueProperty());
```

10.2 Custom converter from the demo

```
StringConverter<Double> converter = new StringConverter<>() {
    @Override public String toString(Double p) {
        if (p == null || p < 0.0) return "Connecting";
        if (p.intValue() == 1) return "Download Complete";
        return String.format("Downloading %.0f%%", p * 100);
    }
    @Override public Double fromString(String text) { return null; }
};
indicator.setConverter(converter);
```

1. Create the indicator.
2. Bind progress to a Service or Task.
3. Choose StyleType from the combo box, as shown in the demo app.
4. Optionally bind startAngle to a slider.

11. Troubleshooting

- If nothing appears, check that the control has non-zero width and height.
- If text is too large, style `.progress-label` or provide a compact converter.
- If the animation keeps running off-screen, make the control invisible; the skin pauses when visibility is false.