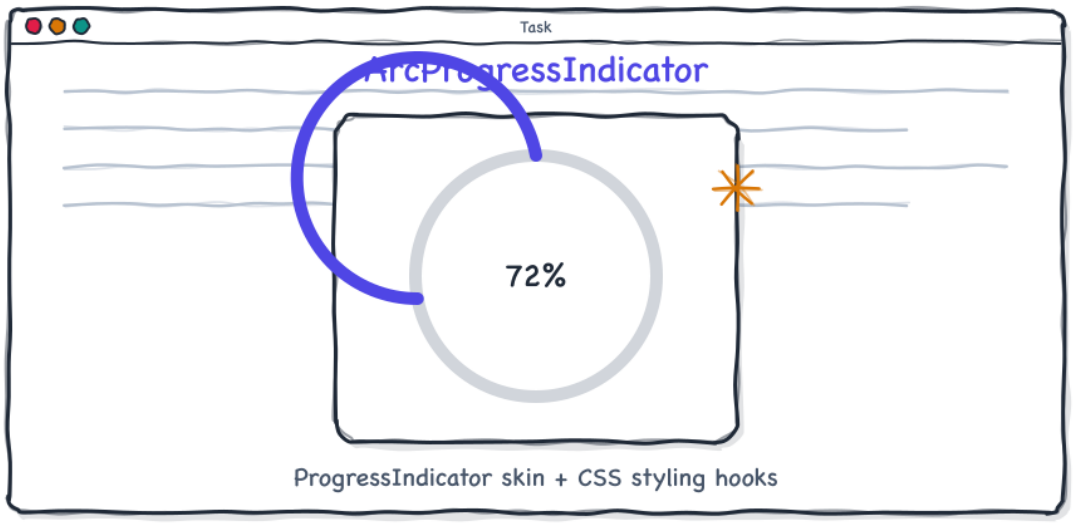
ArcProgressIndicator

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



The shared arc progress model renders a track, progress arc and central label or graphic.

ArcProgressIndicator is the common ProgressIndicator base used by the circle and semi-circle variants. It contributes the shared properties, converter, graphic slot, CSS metadata, pseudo-classes, localization and accessibility behaviour; concrete subclasses provide the actual skin geometry.

Table of Contents

1. Introduction

ArcProgressIndicator extends `javafx.scene.control.ProgressIndicator`. It is declared **abstract**, so applications normally create `CircleProgressIndicator` or `SemiCircleProgressIndicator`. The class is still important because all arc-based indicators inherit its API.

- Determinate progress uses values from `0.0` to `1.0`.
- Indeterminate progress uses the inherited `ProgressIndicator.INDETERMINATE_PROGRESS` value.
- A central label is produced by a `StringConverter<Double>`.
- A custom `graphic` can replace or accompany the label node.
- The `completed` pseudo-class is active when progress is exactly `1.0`.

```
private void configure(ArcProgressIndicator indicator) {
    indicator.setProgress(0.35);
    indicator.setStyleType(ArcProgressIndicator.StyleType.BOLD);
    indicator.setProgressArcType(ArcType.OPEN);
}
```

Configure the shared API on any concrete arc progress indicator.

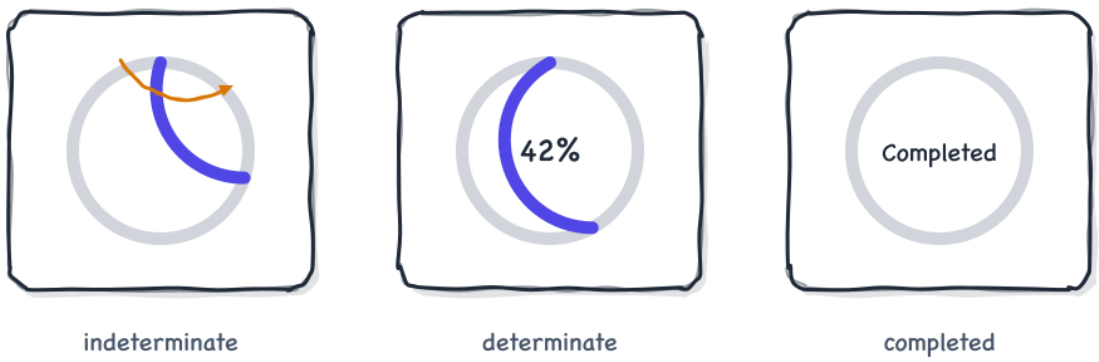
> **Careful.** Use a concrete subclass when creating UI. The base class has no direct default skin of its own.

2. Getting started

Start with a concrete subclass, then configure it through the inherited base properties. Passing `0.0` avoids creating the indeterminate animation during construction.

```
CircleProgressIndicator indicator = new CircleProgressIndicator(0.0);
indicator.setProgress(0.42);
indicator.setConverter(new StringConverter<>() {
    @Override
    public String toString(Double value) {
        return value == null || value < 0 ? "Working" : String.format("%.0f%%", value * 100)
;
    }

    @Override
    public Double fromString(String text) {
        return null;
    }
});
```

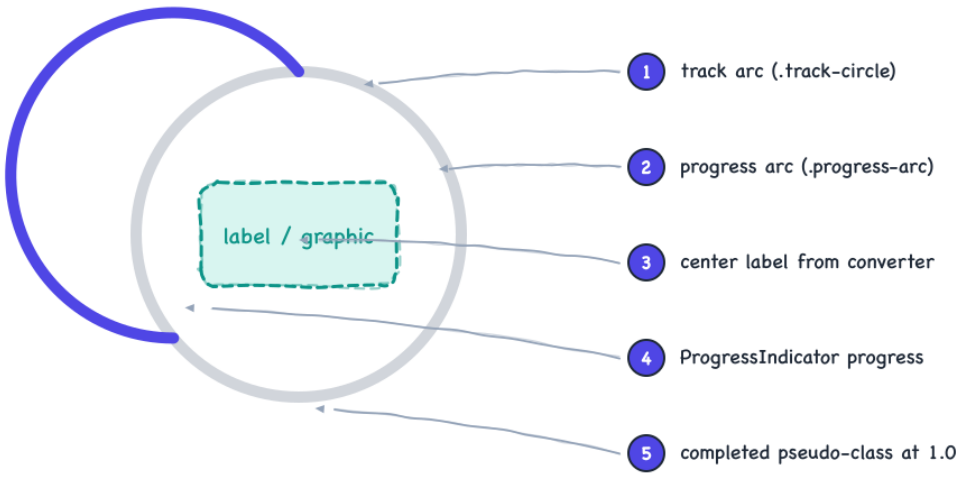


progress < 0 starts the timeline; progress == 1.0 enables :completed

The three progress states inherited from ProgressIndicator plus the completed pseudo-class.

3. Anatomy

The base skin used by the subclasses lays out three children: a track arc, a progress arc and a label. The label binds to both `converter` and `graphic`.



Shared parts contributed by the arc progress base implementation.

Part	Style class	Description
Track arc	`track-circle`	The full background arc; its type is bound to `trackArcType`.
Progress arc	`progress-arc`	The foreground arc; its length is computed from `progress`.
Progress label	`progress-label`	Shows converter text and binds its graphic to `graphic`.

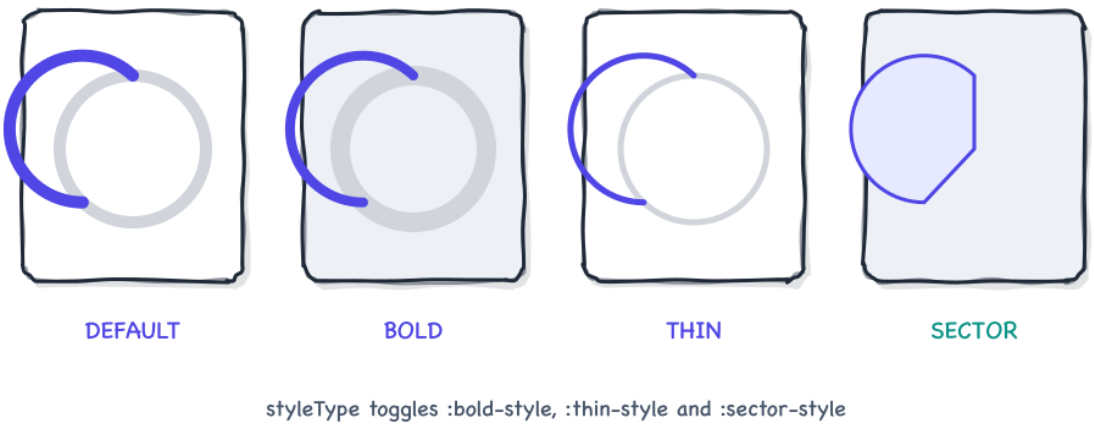
4. Control API

Property	Type	Default	Description
converter	ObjectProperty<StringConverter<Double>>	percent converter	Converts progress to label text. Negative or null progress maps to an empty string; 1.0 maps to localized 'Completed'; other values are floored percentages.
graphic	ObjectProperty<Node>	null	Custom node shown by the label. The label remains visible when a graphic is present.
progressArcType	ObjectProperty<ArcType>	OPEN	Arc type used for the foreground progress arc. Styleable.
trackArcType	ObjectProperty<ArcType>	CHORD	Arc type used for the background track arc. Styleable.
styleType	ObjectProperty<StyleType>	DEFAULT	Visual style: DEFAULT, BOLD, THIN or SECTOR. Styleable and reflected by pseudo-classes.

4.1 Inherited progress

The inherited `progressProperty()` drives the arc length. Values below zero are indeterminate. A value of `1.0` activates `:completed`.

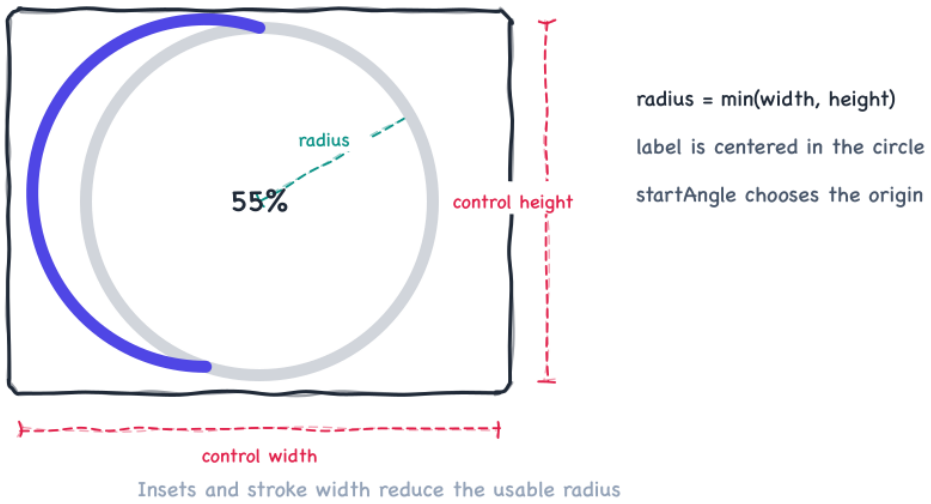
5. Behaviour and states



The StyleType enum changes pseudo-classes that the concrete CSS files style.

State	Condition	Effect
indeterminate	progress < 0	Subclass animation timeline is created lazily and played while visible.
determinate	0.0 .. 1.0	Animation stops and progress arc length becomes maxLength * progress.
completed	progress == 1.0	`:completed` pseudo-class is active and default text is localized.

6. Layout and sizing



Subclasses compute their radius from the available control bounds.

The base skin positions arcs manually in `layoutChildren()`. The concrete skin supplies the radius binding, arc center and label position. Insets and stroke widths reduce the usable drawing radius.

7. Styling

The base stylesheet is `arc-progress-indicator.css`. Concrete controls load their own stylesheets and repeat the same child style classes.

Selector	Purpose
`.arc-progress-indicator`	Root style class inherited by concrete controls.
`.track-circle`	Track stroke; default stroke width 3px, stroke -fx-box-border, transparent fill.
`.progress-arc`	Progress stroke; default stroke width 3px, stroke -fx-accent, transparent fill.
`.progress-label`	Centered text and graphic label.

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

```
.circle-progress-indicator {
    -fx-style-type: bold;
    -fx-progress-arc-type: open;
    -fx-track-arc-type: chord;
}

.circle-progress-indicator .progress-arc {
    -fx-stroke: #22c55e;
}
```

8. Localization

The base class uses `ResourceBundleManager.BundleType.ARC_PROGRESS_INDICATOR`.

Key	English text	Used for
`status.completed`	Completed	Default converter text at progress 1.0.
`accessible.text.loading`	loading	Accessible text while indeterminate.
`accessible.text.percent`	{0} percent	Accessible determinate progress text.

9. Accessibility

The constructor sets `AccessibleRole.PROGRESS_INDICATOR`. Its accessible text is bound to progress until application code supplies its own text. Indeterminate progress reads the localized loading text; determinate progress reads a localized rounded percentage.

```
indicator.setAccessibleText("Synchronization progress");
// Application-set text intentionally takes over from the automatic binding.
```

10. Recipes

10.1 Show domain text

```
indicator.setConverter(new StringConverter<>() {
    @Override public String toString(Double p) {
        return p == null || p < 0 ? "Connecting" : String.format("Downloading %.0f%%", p * 1
00);
    }
    @Override public Double fromString(String text) { return null; }
});
```

10.2 Use a graphic instead of text

```
FontIcon icon = new FontIcon("mdi-cloud-download");
indicator.setConverter(null);
indicator.setGraphic(icon);
```

11. Troubleshooting

- Do not instantiate `ArcProgressIndicator` directly; it is abstract.
- Use `new CircleProgressIndicator(0.0)` when you need no initial animation.
- If a CSS style type appears to do nothing, make sure you are styling the concrete root class as well as the inherited child classes.
- Set progress to exactly `1.0` if you rely on `:completed`.