Dependency Inversion

Contents

Overview

In object-oriented programming, the last of the **SOLID Principles** is **D** - which stands for **Dependency Inversion**.

The *Dependency Inversion Principle* states two key points:

- *high-level modules should not depend on low-level modules* - both should depend on *abstractions*, such as interfaces
- *abstractions should not depend on details* (concrete implementations) - instead, *details should depend on abstractions*

Essentially, both higher- and lower-level modules should depend on the same abstractions, rather than on each other.

Dependency Inversion in Action

WeatherTracker

Let's unpack this by looking at a program containing three classes:

- `WeatherTracker.java`
- `Email.java`
- `Phone.java`

▶ Weather Tracker
▶ Email
▶ Phone

Currently, this does not adhere to the *Dependency Inversion Principle*, because the high-level module `WeatherTracker.java` depends on the low-level details of the different notification systems we have in `Email.java` and `Phone.java`.

Fixing the Weather Tracker

Since we want both the higher- and lower-level modules to depend on the same abstraction, we'll use a `Notifier.java` interface to do this:

▶ Notifier

We can now slightly edit the lower-level modules so that they implement the `Notifier.java` interface:

▶ Email
▶ Phone

We can now add a function to call to the `Notifier.java` interface from inside the higher-level `WeatherTracker.java` module:

▶ WeatherTracker

Tutorial

There is no tutorial for this module.

Exercises

Let's say that you're working on a big project. The project has both backend and frontend developers within it.

Consider the following program, which simulates a development team and the project they are working on:

- `BackendDeveloper.java`
- `FrontendDeveloper.java`
- `Project.java`

▶ BackendDeveloper
▶ FrontendDeveloper
▶ Project

This current setup violates the *Dependency Inversion Principles* because the high-level `Project.java` module depends on both the `BackendDeveloper.java` and `FrontendDeveloper.java` modules.

Refactor the program using the following four modules to ensure that it adheres to the *Dependency Inversion Principle* to complete this exercise:

- `BackendDeveloper.java`
- `FrontendDeveloper.java`
- `Project.java`
- `Developer.java` (interface)

▶ See solution