**S‑Emulator - Assignment 1 (+bonus 1 implementation)**

**Submitters:**

* **Shir Luzon**, 208299057, luzon.shir@gmail.com
* **Noa Dinbar**, 211821525, noadinbar@gmail.com

**Github repository:** <https://github.com/noadinbar/S-emulator.git>

**Overview**

S‑emulator, split into three modules: engine (core logic, passive), dto (deep data transfer objects), and ui (console app). UI handles all input/output and formatting.

**Architecture & Dependencies**

• ui depends on engine and dto.

• engine depends on dto.

• dto provides deep objects so that the UI never touches engine internals.

**Main packages (package‑level)**

***engine module***

Core emulator logic: program model, instruction set (basic/synthetic), execution, validation, and degree‑expansion utilities.

1. **api** - Interfaces (LoadAPI, DisplayAPI, ExecutionAPI) used by the UI and XMLLoader for loading the xml file.
2. **exportToDTO** - API implementations and mapping from engine internals to DTOs.
3. **structure** - Program, instructions (basic/synthetic), variables, labels, also subpackages for execution and expansion.
4. **exceptions** - Focused exceptions for validation and input errors.

***dto module***

For UI↔engine communication.

1. **display** - DTOs for presenting the program (name, inputs/labels used, instruction list in the required format).
2. **execution** - DTOs for runs (request with degree+inputs, result with y/final variables/cycles/executed program), and history entries.
3. **types** - Shared value types (variables/labels/options).

***ui module***

Console application: entry point, menu loop, command screens, and all formatting/output.

1. **app** -Main entry point and the console loop.
2. **menu** - Menu items (Load XML, Display, Expand, Execute, History, Save, Load, Exit).
3. **screens** - Action handlers per command (call APIs, validate input, print output).
4. **format** - Instruction/variables/history formatting.

**Current capabilities**

1. **Load XML** - Parse and validate; a valid load replaces the previous program, an invalid load leaves the previous one intact.
2. **Display Program** - Show program name, inputs used, labels used (EXIT last), and instructions in the required format.
3. **Expand** - Expand synthetic instructions up to a chosen degree and show their origin chain.
4. **Execute** - Choose degree (0=As‑Is), supply inputs (CSV), run, and display: y, all variables (y → xᵢ → zᵢ), and cycle count.
5. **History** - Keep past runs: index, degree, inputs, final y, cycles.

**Bonus — Implemented**

We implemented save\load with Java serialization:

1. **Save** - Saves the current Program (including run history) to a .ser file with a requested path and name.
2. **Load State** - Loads a previously saved Program back into the app.

**Build & Run**

**Artifacts**: three JARs — ui.jar, engine.jar, dto.jar (packaged in the distribution).

**Run**: Double‑click run.bat.