FDMJ2024 SLP Specification

The grammar

```
Program -> MainClass
MainClass -> class main '{' Statememt* '}'
Statement -> id = Exp ';' | putint '(' Exp ')' ';' | putch '(' Exp ')' ';'
Exp -> Exp op Exp | id | '-' Exp | INT_CONST | '(' Exp ')' |
        '(' '{' Statement* '}' Exp ')'
```

Notes:

The semantics of an FDMJ2023 SLP program with the above grammar is similar to that for programming languages of C and Java. Here we give a few notes about it, and we will have more discussions during the semester.

- The root of the grammar is `Program`.
- The binary operations (`op`) are `+`, `-`, `*`, `/`
- `INT_CONST` is `[0-9]+`.
- `id` is any string consisting of `[a-z]`, `[A-Z]`, `[0-9]` and `_` (the underscore) of any length, with the restriction that it cannot be any of the keywords (terminal strings marked red) used in the above grammar, and it must start with a `[a-z]` or `[A-Z]`. The lower or upper case letters in an id are significant (e.g., `aB` and `ab` are two different ids).
- All the statements are executed from left to right, including the ones in the escape expression, and only impact the state after the point of the code. For example, if the initial value of `a` is `0`, then `a+2*({a=1; b=2} a+b)` gives `6`, but `2*({a=1; b=2} a+b)+a` gives `7`.