Class **Operator**

- string name;
- string expression;
- virtual string expressionEval();
- int toInt();
- string toString();

Class **AssignmentVar** : public Operator

- virtual string expressionEval();

Class **AssignmentFunction** : public Operator

- string real_param;
- virtual string expressionEval();
- virtual string realToFaktParam();

Class **Print** : public Operator

- virtual string expressionEval();

1. Всеки един ред от текстовия файл се записва като обект от съответния му клас.
2. Ред “print <expression>” се записва като обект от тип Print с име “print”, а <expression> се оценява посредством expressionEval() ф-цията, която е същата като при класа **AssignmentVar**.
3. Ред “read <var>” се записва като обект от тип **AssignmentVar** с име <var> и съответно expression – въведен от потребителя.
4. Разликата между expressionEval() на **AssignmentVar** и **AssignmentFunction** класовете е в това, че при дефиниране на променлива изразът й се оценява директно, а при дефиниране на функция – в тялото на функцията се проверява дали има невалидни променливи (недефинирани досега), без да оценява тялото, т.к. функцията няма зададен фактически параметър.
5. При оценяване на израз (за **AssignmentVar** и **Print**), ако в израза има извикване на функция:
    a. Проверява се дали има вече дефинирана функция с такова име (т.е. търсим обект с името на функцията);
    b. Ако съществува такъв обект: извикваме неговата функция за заменяне на реалния параметър на тялото на тази функция с фактическия, подаден в израза;
    c. Веднъж когато имаме запаметен новия израз в променлива, извикваме рекурсивно expressionEval() на текущия клас (съотв. **AssignmentVar** или **Print**) и така получаваме оценката на извиканата функция.
6. При “print <expression>” <expression> се оценява на момента и се запазва в обекта; Накрая след като всички редове са прочетени, с цикъл обхождаме всички обекти и извеждаме <expression> на тези, чието име е “print”.

Пояснение 1: Няма клас **Read**, т.к. “read a” се запазва като обект от тип **AssignmentVar**.
Пояснение 2: След оценката на всеки <expression> се проверява какво връща той; Ако оценката е NULL => има грешка в <expression> и извежда съобщение за грешка.