نکته‌ی استفاده از idtable.add_string("something") اینجوریه:

کلا این add_string میاد اون "something" رو به idtable اضافه می‌کنه و
returns a type derived from Entry to describe the symbol table entry.

حالا توی COOL وقتی یه چیزی optional عه و مهم نیست که کلا no_exp می‌ذاریم. ولی دو جا هست که چیزای optional امون اگر تعریف نشن یعنی یه مقدار دیفالت دارن که اون دیفالت رو ما باید واسه پارسر مشخص کنیم .

حالت اول:

constructor واسه class به فرم:

```
constructor class_(name : Symbol; parent: Symbol;
                   features : Features; filename : Symbol): Class_;
```

هست. و می‌دونیم که

Each constructor takes typed arguments and returns a typed result. The types may either be phyla or any ordinary C++ type.

یعنی این constructor داره یه Symbol یا همون type identifier واسه ورودی parent می‌گیره و builds a Class tree node with the four arguments as children.

حالا حالتی رو داریم که class هایی هستند که parent اشون مشخص نشده! یعنی دارن از object ارثبری می‌کنند. توی این حالت باید idtable.add_string("Object") رو به جای parent اینا بذاریم. که یه type identifier (یا همون Symbol) برمی‌گردونه که میره توی ورودی class_ به جای parent اش می‌شینه.

حالت دوم:

constructor واسه dispatch به فرم:

```
constructor dispatch(expr : Expression;  name : Symbol;
                     actual : Expressions) : Expression;
```

هست. حالا وقتی یه method call یا همون dispatch داریم که توش expr اولیه نیومده؛ یعنی باید جاش self بذاریم! و این یعنی باید idtable.add_string("self") رو صدا بزنیم! ولی این بهمون یه type identifier (یا همون Symbol) برمی‌گردونه ولی چیزی که ما می‌تونیم جای expr بذاریم از تایپ یا به بیان اینا phyla عه Expression هست. پس با استفاده از constructor :

```
constructor object(name: Symbol): Expression;
```

ما این Symbol دریافتی رو می‌دیم به object و اون برامون یه tree node می‌سازه از type/phyla عه Expression هست که می‌تونیم بدیمش به dispatch.